US Army Corps
of Engineers®
Engineer Research and
Development Center

ERDC-CERL LD-870 Download Program

Programming Manual

Ben Niemoeller and Edward T. Nykaza   May 2007

Construction Engineering
Research Laboratory

ERDC-CERL LD-870 Download Program

Programming Manual

Ben Niemoeller and Edward T. Nykaza

Construction Engineering Research Laboratory
U.S. Army Engineer Research and Development Center
PO Box 9005
Champaign, IL 61826-9005

Final report

Approved for public release; distribution is unlimited.

Prepared for U.S. Army Aberdeen Test Center
Aberdeen Proving Ground, MD 21005-5059

Under Work Unit 0B72D9

Abstract: The U.S. Army Engineer Research and Development Center Construction Engineering Research Laboratory has developed software that interfaces with an array of Larson-Davis Model 870 Environmental Noise Monitors for Aberdeen Test Center. This document explains logic and procedures used while programming the software that are of interest to a programmer looking to modify or expand the functionality of the program. The following topic areas are covered: terminology, time synchronization, and scheduling events. This document will be of interest to those who wish to modify the L-D Download software program. The code, which was written with Microsoft Visual Studio 2005, is included in the appendices.

Contents

Preface

This study was conducted for Aberdeen Test Center (ATC), Aberdeen Proving Ground, MD, under MIPR6FXXR3A563, "R&D of Military Noise Assessment Tools to Support ATC's Noise Program." The ATC technical monitor was Kimberley Fillinger.

The work was performed by the Ecological Processes Branch (CN-N) of the Installations Division (CN), Construction Engineering Research Laboratory (CERL). The CERL Principal Investigator was Edward T. Nykaza. Alan B. Anderson is Chief, CN-N, and Dr. John T. Bandy is Chief, CN. The associated Technical Director was William Severinghaus, CVT. The Deputy Director of CERL is Dr. Kirankumar V. Topudurti and the Director of CERL is Dr. Ilker Adiguzel.

CERL is an element of the U.S. Army Engineer Research and Development Center (ERDC), U.S. Army Corps of Engineers. The Commander and Executive Director of ERDC is COL Richard B. Jenkins, and the Director of ERDC is Dr. James R. Houston.

1   Introduction

Background

Aberdeen Test Center (ATC) in Maryland has one of the top installation
environmental noise programs in the nation. The reason they are at the
forefront of environmental noise programs is largely due to the trained
staff and number of noise monitors they have located in the communities
surrounding the installation. Over the past several years, there have been
issues with the reliability of the software used to download data from the
noise monitors.

In April 2006 ATC funded the U.S. Army Engineer Research and Devel-
opment Center Construction Engineering Research Laboratory
(ERDC/CERL) to replace the outdated software ATC uses to download and
manage the data from their noise monitors. The new software developed
by ERDC/CERL was built upon software developed for a research project
being conducted at Aberdeen Proving Ground and modified to meet the
needs of the ATC staff who run the system on a day-to-day basis. The soft-
ware went into operation in August 2006.

Objectives

The objective of the work was to develop a Windows-based server and da-
tabase software to download data from ATC's environmental noise moni-
tors.

Approach

This document contains ERDC/CERL's knowledge of the noise monitoring
system at Aberdeen Proving Ground. The document explains some of the
logic and procedures used to program the software. It covers the following
topic areas: terminology, unit list, time synchronization, scheduler, and
Larson Davis (LD) unit class variables. The code, which was written with
Microsoft Visual Studio 2005, is included in the appendices.

Scope

This report contains the software code and logic for the noise monitoring
system and is intended to be used by programmers who wish to update the

software. A separate report (ERDC/CERL SR-07-6) contains operational knowledge and is intended to serve as a user's manual.

2 Terminology

This terminology is used to define the structure, hierarchy, and functions of an object-oriented program, in addition to items that are specific to .NET and Windows programming.

Namespace – a level of hierarchy above class. In Visual Basic (VB), all of the code for one project resides in a single namespace. The namespace for the Aberdeen Test Center download program is WindowsApplication1. The namespaces for other programs are L_D_Admin, L_D_Download_1_modem, and Configurator.

Class – A class is a logical grouping of data, along with instructions to perform on that data.

Object – An instance of a class, generated at runtime. When code mentions a specific instance of a class, that instance is often called an object.

Object data type – In .NET, an Object data type is also the most primitive data type; that is, all other data types are inherited from the Object data type.

Field – A field is a variable inside the class that is accessible by all methods of that class. This is called a "class variable" outside the realm of Windows programming. A field can contain a primitive data type or an instance of a class.

Method – The executable code within a class is contained in one or more methods. A method can call other methods, access data in its own class as well as in other classes, and even instantiate new classes. VB has two kinds of methods, a Sub(routine) and a Function. A Sub is a method that does not explicitly return data to the method that called it. A Function must return some data, via a Return statement, to the method that called it.

Local variable – a variable declared inside a method. This is often just called a "variable." The variable, as well as the data contained in it, is lost when the method exits.

Block – the code contained within an If…Then statement or a program loop. The code in a block will be indented.

Block variable – a variable declared inside an If…Then block or a loop. The variable, as well as the data contained in it, is lost when the end of the programming block is reached.

An example program hierarchy:

```
Namespace
     Class
           Field
           Field
           Field
           Method
                 Local Variable
                 Local Variable
                 If X then
                       (block 1)
                 Else
                       (block 2)
                 End If
           End Method
     End Class
End Namespace
```

Constructor – A special method in a class which executes when a new instance of that class is created. A constructor is used to write the "default" values to the fields in a class. The constructor for a Form is hidden away in the .Designer file for that form.

Property – A special method in a class which is used explicitly to store and retrieve data from a field. If a field must be accessed from outside of the class containing the field, a property must be written for the field, and called upon to store and retrieve the field's data.

Data type – When you declare that a field or variable will contain a String, the variable is then said to be "of data type String" or "of type String". Similarly, if you were to declare that a variable will hold an instance of the Shoe class, that variable is said to be "of type Shoe." The consequence of data typing is that once you declare that a variable will hold one type of data, you cannot later on in the program assign another type of data to this variable. This prevents runtime errors due to an operation on

incorrectly formatted data, such as attempting to add a number to a piece of text.

Static (Shared) vs. **Instance** – For a more thorough definition, see pp. 257-266 of the Balena VB book. In summary, fields and methods can be declared either static by using the Shared keyword, or as instance members by omitting the Shared keyword. Static methods cannot access instance fields, and caution should be used when accessing static fields with instance methods. A class cannot be declared Shared. However, the static members (fields and methods) within a particular class are accessible at anytime by any class, without needing to refer to a specific instance of the class created in the program.

Use instance fields to hold all program data that you might need to modify later on.

Form – Called a "window" by the user. In .NET, a form is represented by a class, and that class is split into two files. For a form named Form1, the file Form1.vb holds code written by the programmer. The file Form1.Designer.vb holds code written by Visual Studio which describes the window's appearance. The .Designer file is not normally accessible from within Visual Studio, and should not be hand-modified by the programmer. Any changes you make to the .Designer file will be overwritten the next time you modify that form's control surface.

Control – The name for things in the window that the user can click on. They are called "controls" because they allow the user to control program flow and execution. Each control is contained in a class.

3   About the Unit List

The Unit List, labeled `LDUntList` in the source code, is managed by a checkout system. This checkout system consists of the `GetUnit` and `ReturnUnit` methods. When a part of the program needs to access a Unit (an object of type `LDUnit`), it makes a call to `GetUnit`. The calling method can ask `GetUnit` to do one of two things: retrieve the first available unit in the list which has not been downloaded, or retrieve a specific unit according to its ID number (`UnitID` or L-number). If `GetUnit` succeeds in finding the requested unit, it will *remove the unit from* `LDUnitList` and pass it back to the calling method. If `LDUnitList` were a library, `GetUnit`'s function would be to find and check out books to a patron. When the calling method is done with the unit, it returns the Unit to `LDUnitList` by calling `ReturnUnit`. `ReturnUnit` simply adds the Unit to the end of `LDUnitList`.

The checkout system ensures that only one section of the program can access a unit at any one time. This safeguard prevents a unit from being accessed by two modems simultaneously. For example, suppose that modem 1 tries to dial unit 9 at the same time as unit 9 dials into, and connects to, modem 3. Without a checkout system, modem 3 would write valid status information and report data to the Unit that represents unit 9, while modem 1 would write status information to the Unit indicating that the program could not connect to the unit. This possibility of data corruption and/or loss is unacceptable for a program whose mission is to gather data for a research project.

Under the checkout system, either modem 1 or modem 3 would grab exclusive, guaranteed access to the Unit. If modem 1 had access, the call on Modem 3 would be dropped and unit 1 would dial until it reached the (eventually) free modem. If modem 3 grabbed access before modem 1 could, the checkout system would give modem 1 another Unit to call while modem 3 downloaded and stored data from Unit 9.

To maintain the integrity of the queuing system, it was found to be necessary to remove *all* of the Units from `LDUnitList` when the user opened Manual Dial (and later Unit Options), and send them to the Manual Dial (`OneUnitDialog`) or Unit Options (`UnitOptions`) classes. When the user closed either of these windows, the Units were returned, with changes

to some of the variables in each Unit, to `LDUnitList`. Taking the units away from other parts of the program ensured synchronization between the data displayed on screed and the data stored in the Units. The downside, of course, is that the four modems cannot obtain any new Units from `LDUnitList`. If a physical unit calls in, it cannot be downloaded. If a modem is already connected to a unit, it can finish that download, but then no more units will be called until Unit Options or Manual Dial are closed.

It was decided to use the checkout mechanism early in the program's development, before becoming familiar with the monitoring *system* as a whole and how the staff at the Aberdeen Test Center (ATC) uses this system. Today there are many situations when it would be advantageous to always keep a complete list of units in memory, and for Unit Options to be able to access a unit at the same time a modem can. It was thought that there would be a need to prevent Unit Options and the modem from writing data to the same variables at once. As it worked out, Unit Options writes to variables that are not written to by the modem, and vice versa. Furthermore, the telephone system permits only one modem to connect to one unit at any given time. Since the download instructions do not execute until the modem is connected to a unit, the rare condition where two modems would try to download a Unit (and overwrite Unit measurement data in the process) is eliminated. It was also thought the program would spend most of its time dialing out to units; it was not anticipated that the program would be used as a server for units calling in. Now the server must be stopped every time the user wants to open Unit Options.

The Unit Viewer is a workaround to not being able to leave Unit Options open indefinitely while the program is running. Unit Viewer gets its data by making a copy of `LDUnitList` and reading the data from the copy. A new copy of `LDUnitList` is sent to Unit Viewer every time `GetUnit` or `ReturnUnit` is called. Unit Viewer compares the new list in memory to the copy of the list stored on disk in order to figure out which units are being downloaded at the moment and display the data appropriately.

Certainly a nondestructive queuing mechanism for `LDUnitList` could be implemented. `GetUnit` could be modified to return the address of the desired Unit on `LDUnitList`, rather than the Unit itself. One would need to use a flag (a Boolean variable) to indicate when a particular unit on the list is being accessed by a modem, so that other units will not grab it. When writing data to a unit, one would need to take care to write only to the

variable in question, and not attempt to place an entire unit back on the list.

For example, a modem wants to set `IsDownloadInProgress` and `Is-DownloadDone` to False. In the current system, the modem would remove a unit from `LDUnitList`, write to the variables within the unit, then write the entire *unit* back to `LDUnitList`. The code looks something like this:

```
Public Sub Foo() 'some method

    'remove unit 9 from LDUnitList
       Dim unit as LDUnit() = GetUnit("L09")

       'if unit doesn't exist, exit sub before the invalid unit is
       'pushed back onto the LDUnitList
       If unit.UnitNum = "null" Then
             Exit Sub
       End If

       unit.IsDownloadInProgress = False
       unit.IsDownloadDone = False
    ReturnUnit(unit)
End Sub
```

To write the same data without using the checkout system, use this code snippet:

```
Public Sub Foobar() 'another method

    'get the index of unit 9
    Dim idx as Integer = GetUnitIndex("L09")

    'Use Synclock whenever writing to a shared resource, such as
       'LDUnitList. SyncLock is not necessary when reading from the
       'list.

       'Use a Try…Catch block to catch the exception that will be
       'thrown if GetUnitIndex returned an invalid index. Note that
       'this strategy is more flexible than exiting the entire
       'method.

       Try
       Synclock Form1.LDUnitList
          With Form1.LDUnitList.Item(idx)
              .IsDownloadInProgress = False
              .IsDownlaodDone = False
```

```
        End With
      End Synclock
    Catch ex As Exception
       'no need to do much here
    End Try
End Sub
```

`GetUnitIndex` is a method similar to `GetUnit`, only instead of returning an actual unit, it returns the unit's position (or address, or index) on `LDUnitList`.

Another issue that remains with nondestructive list management is synchronizing the data displayed in the Manual Dial and Unit Options windows with the current data in `LDUnitList`. An effort will need to be made to keep the function of the Cancel button intact, since it will be convenient to write directly to `LDUnitList` as the user clicks around on the window surface. User Interfaces in general are finicky, and much testing will need to be done to ensure the user can't screw up the download process by changing a unit's options, especially since the wall between Unit Options and the modems will no longer exist.

Perhaps the biggest challenge lies in testing the system. The current list management system has been in use for 6 months, and its strengths and weaknesses are fairly well known. Any new system will need to be tested for many weeks alongside the current system. This will be hard to do without getting extra telephone lines and modems, as testing a queuing system with one modem will not reveal much about that system. It will be a challenge and was not in the scope of this project.

A final note: Whatever queuing system you use, the download and storage of report data will not be affected. Reports are downloaded then immediately checked for errors and written to the SQL database. After the report is written to the database or to disk, the report data are flushed from the Unit, and thus from `LDUnitList`.

4   Time Synchronization

Time synchronization of the units is of critical importance to the noise monitoring system. Without time synchronization, it is impossible to discriminate a blast event from wind noise. Also, it would be impossible to determine an event's origin, which is important for research purposes. L-D Download finds the difference between the clock on each unit and the clock on the computer, and attempts to write a corrected time stamp to each exceedance and interval record using this synchronization data.

Due to the high latency of modem connections, L-D Download does not assume that the 870's clock can be set to the same time as the computer's clock. Instead, L-D Download reads the 870's clock and finds the difference between the 870's clock and the computer's clock. This is referred to as **time offset** in the code. Furthermore, L-D Download tracks the change in the *offset* between the unit and computer over a period of time. This is done to account for drift in the 870's internal clock, and is referred to as **time drift** in the code.

The first step to time synchronization is getting an accurate reading of the unit's clock and determining how much it differs, at that instant, from the host computer's clock. The unit and computer times are read using the `GetUnitTime` function in the Modem class (line 2507 in the Modem.vb file). `GetUnitTime` sends an R2 command to the unit, then grabs the computer's time and stores it to a cell in the `hostDate` array. The critical latency here is the time that passes between when the R2 command is sent and when it is received by the unit. If the computer time is grabbed at the exact second the unit *sends* its response, then the time offset between the computer and unit is accurate. The unit is assumed to send its response immediately after it receives the R2 command. At minimum, the unit will send a response a fixed amount of time after receiving the command; this time can be measured and accounted for by adding the delta to the computer time or else subtracting it from the unit time. Currently, the program adds 45 milliseconds to the stamped computer time to account for a minimum send latency of 45 ms. When the computer receives the response from the unit, it parses the text into a Date object and stores it to a cell in the `unitDate` array. If a response is delayed more than five seconds, the program will not store a response to the `unitDate` array. This

error will be caught later on and prompt the program to take more samples of the unit's time.

When using modems, the latency can vary unpredictably, even during the same connection session. When calculating time drift, it is important that a unit that was called on Tuesday have the same latency when called on Wednesday or Thursday, even if the exact latency on Tuesday isn't known. To make the time measurements less susceptible to variations in latency, the unit and computer clocks are polled eight times. For each pair of unit and computer time stamps, the difference (offset) between them is calculated and stored to a cell in the `tDifferenceTemp` array. It is worthy to note that Windows stores a time stamp as a 64-bit integer representing the number of 100-nanosecond intervals that have elapsed since the year 0 A.D. Once the eight unit and computer time stamps are taken, L-D Download computes the numeric average of the stamps by shifting each time stamp three bits to the right (to perform a bitwise division by eight), then adding the shifted stamps together. The bitwise division is used to prevent overflow errors, at the expense of 200 ns' worth of time precision. The averaged unit time is stored in `unitDateTicks` while the averaged computer time is stored in `hostDateTicks`. The average offset is taken by multiplying each of the offsets stored in the `tDifferenceTemp` array by 0.125, then adding them together and storing the result in `tOffsetAvg`. If the time data is valid (no missing time stamps in the `unitDate` and `hostDate` arrays due to a receive timeout), the averaged time data is returned to the calling method as a `ReferenceTime` object.

The second step to time synchronization is determining when each exceedance took place with regards to the computer's clock. To do this, one needs to know two things: the offset between the unit and computer at a previous point in time, and the offset at a time on or after the event took place. For example, suppose that the computer dialed the unit at 0600 and observed that the unit's clock was three seconds ahead of the computer's clock. Suppose that the computer dials the unit again at 1800 hours, only this time observes that the unit is nine seconds ahead of the computer. Assuming that no one tampered with the unit's clock, one may surmise that between 0600 and 1800, the unit's clock was between three and nine seconds ahead of the computer's clock. This means any event that took place at a time between 0600 and 1800, according to the unit, took place between 3 and 9 seconds earlier, according to the computer.

So, when exactly did the event take place? If you can assume that the unit and host clocks drifted in a predictable fashion, you can estimate the offset between the unit and host at the time of the event using offsets taken before and after the event occurred. <u>L-D Download assumes that the unit clock drifts linearly over time, and the computer clock does not drift at all</u>. Using this assumption, if the offsets at a point in time before and after the event are known, then the offset for any event that occurred between these endpoints can be calculated precisely using the equation

$$O_e = O_i + \frac{O_f - O_i}{T_{fh} - T_{ih}} \cdot \left(T_{eu} - \left(T_{ih} + O_i \right) \right) \qquad (1)$$

where:

O_e = offset between unit and host at the time of event
O_i = offset at a known point in time preceding the event
O_f = offset at a known point in time following the event
T_{eu} = time the event occurred, according to the unit
T_{ih} = computer's time at the known point in time preceding the event
T_{fh} = computer's time at the known point in time following the event

Putting equation 1 in terms of unit and host times, the time at which the event occurred according to the computer's clock is:

$$H_e = U_e + \left(H_i - U_i \right) + \frac{\left(H_f - U_f \right) - \left(H_i - U_i \right)}{H_f - H_i} \cdot \left(U_e - U_i \right) \qquad (2)$$

where:

H_e, U_e = host (computer) and unit clocks at the time the event occurred
H_i, U_i = host and unit clocks at the known point in time preceding the event
H_f, U_f = host and unit clocks at the known point in time following the event

When the unit's time is reset, L-D Download will grab the unit and host times and store them to disk as a serialized ReferenceTime object, using

the `SerializeReferenceTime` function in the `LDUnit` class. The program stores one reference time for each unit in the `ReferenceTimes` subfolder of the program folder, using the unit number as the filename. This stored time serves as the known point in time preceding the event; the program will not attempt to append corrected times to any event whose time stamp occurs before this time. The known point in time following the event is the time grabbed when the unit is called, just before a download begins.

The time correction calculation starts at line 1592 in the Modem class file (Modem.vb). For each event and interval read from a unit, L-D Download reads the time stamp set by the unit, then solves equation 2 to determine when the event or interval occurred according to the computer. In the code, H_i and U_i are the variables `RT22.HostTimeZero` and `RT22.UnitTimeZero`, respectively, while H_f and U_f are the local variables `hostDateAvg` and `unitDateAvg`. The difference between the unit and computer event times is O_e from equation 1, which in the code is represented by the block variable drift on line 1619. The word drift is a misnomer, since this variable contains an offset value that is added to the event's time stamp given by the unit. Finally, a corrected time stamp is appended to the end of the event or interval record as a text string.

In a real system, one can assume linear drift if (a) the unit's oscillator (from which the clock is derived) does not change frequency with the temperature variation seen over the course of a typical day, and (b) the computer's clock is continuously synchronized to an stable time source such as the National Institute of Standards and Technology (NIST) Internet time service. Currently, the noise computer is synchronized to NIST once per hour using a software program called NISTIME. Furthermore, Windows Time Service on this machine is disabled; the time service attempted to synchronize the computer to the ATC domain controller, whose clock showed a drift of 1 second per hour during testing in August 2006.

5   Scheduler

When L-D Download is started, a Timer object named `Timer1` is created. The purpose of `Timer1` is to fire an event called Timer1.Tick once each second. A method in Form1 called `Timer1_Tick` handles the Tick event. This `Timer1_Tick` method (line 775 in Form1.vb) and its helper methods `WorkdaySched` and `WeekendSched` (lines 891 and 1065, respectively) form the core of the Scheduler.

Each time Timer1_Tick runs, it grabs the computer time and rounds off the fractional seconds. Next, it looks at the time of day and sets the `Is-Daytime` flag accordingly, then runs either `WorkdaySched` or `Weekend-Sched` depending on the day of the week. `WorkdaySched` and `Weekend-Sched` use a `Select Case` statement to see if it needs to run a scheduled task at this point in time. If the computer's time matches one of times in a `Case` statement, the code beneath that `Case` statement will run. If it does not match a time in the `Case` statement, the code execution will jump to the end of the Select block.

To add an event to the Scheduler, use the following template:

```
'Select Case nowtime 'already in code, nowtime is a TimeSpan
object representing the current computer time

    Case <TimeSpan> 'time at which you want the event to occur,
represented by a TimeSpan object. To schedule an event for 2pm,
you can use "New TimeSpan(14, 0, 0)" or use a class variable that
has an equivalent TimeSpan value.

        Dim uList As List(Of LDUnit) = GetAllUnits() 'grab all
units from LDUnitList

        For Each unit As LDUnit In uList
            'Usually you will have different settings for the
Aberdeen
            'and CERL units. Set the variables for each unit
below.
            'Make sure to set IsDownloadDone and
IsDownloadInProgress
            'to False in order to download the unit.
            If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
                unit.AllowCallIns = True
                unit.ResetDataYN = True
                unit.ResetTimeYN = True
                unit.DLTries = 0
                unit.IncludeLC = False
                unit.IncludeQ = False
```

```vbnet
                unit.IncludeR = False
                unit.IsDownloadDone = False
                unit.IsDownloadInProgress = False
            Else
                unit.AllowCallIns = False
                unit.ResetDataYN = True
                unit.ResetTimeYN = True
                unit.DLTries = 0
                unit.IncludeLC = False
                unit.IncludeQ = False
                unit.IncludeR = False
                unit.IsDownloadDone = False
                unit.IsDownloadInProgress = False
            End If
        'desired exceedance threshold is now determined in the
Modem.Download method
        Next

        ReturnAllUnits(uList) 'return all units to LDUnitList

        'Select which modems to use to call the units – a False
indicates that the modem will dial out, a True indicates that the
modem will be monitoring for incoming calls

        CheckBox1.Checked = False
        CheckBox2.Checked = False
        CheckBox3.Checked = False
        CheckBox4.Checked = False

        'Finally, call the units
        If Not Me.BWDelayedClick.IsBusy Then
            Me.BWDelayedClick.RunWorkerAsync()      'dial out
        Else
            Dim eaM As New
StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingTex
t)
            eaM.PrecedingText = "Cannot start download at " &
Date.Now.ToString("M/d/yyyy h:mm:ss tt") _
            & "due to a busy worker thread." & vbLf & vbLf
            RaiseEvent UpdateMessages(Me, eaM)
        End If

    'End Select
```

6   LDUnit Class Variables

The class variables in LDUnit are accessed through their corresponding properties. The variables themselves are marked Protected, while the properties are Public.

`Public Property InstallPath() As String`

> The base path of L-D Download. By defalut, this is C:\Program Files\L-D Download.

<u>These properties are set by the user, and tell L-D Download how to download the unit.</u>

`Public Property IncludeR() As Boolean`

- True – L-D Download will retrieve the Read variables from the unit.
- False – L-D Download will not retrieve the Read variables from the unit.

`Public Property IncludeLC() As Boolean`

- True – L-D Download will retrieve the Log and Cal variables from the unit.
- False – L-D Download will not retrieve the Log and Cal variables from the unit.

`Public Property IncludeQ() As Boolean`

- True – L-D Download will retrieve the Query variables from the unit.
- False – L-D Download will not retrieve the Query variables from the unit.

`Public Property AllowCallIns() As Boolean`

- True – L-D Download will tell the unit to call the noise computer when an exceedance or alarm is recorded (sets Q??? to 3). L-D Download will also enter the noise computer's phone number into the unit.

- False – L-D Download will tell the unit to never call a computer.

`Public Property` ResetDataYN() `As Boolean`

- True – L-D Download will issue a reset-all S1,1 command to the unit. All measurement data on the unit will be erased.
- False – L-D Download will not issue a reset-all command.

`Public Property` ResetTimeYN() `As Boolean`

- True – L-D Download will set the clock on the unit, determine the offset between the unit and computer clocks, and record the offset to disk. Currently the program does not set the date or the day of week on the unit.
- False – L-D Download will not set the clock on the unit.

`Public Property` IsDownloadDone() `As Boolean`

- True if L-D Download completes the download procedure success-fully, False if not. Used to determine if a unit needs to be dialed again.
- This variable is only used when a dial command has been issued by the scheduler or user. If a unit calls in and its download fails, that unit will not be re-dialed by the program.

`Public Property` IsDownloadInProgress() `As Boolean`

- True if the unit is currently being downloaded, False otherwise.

`Public Property` IsEnabled() `As Boolean`

- True – When a dial command is issued, L-D Download will dial this unit.
- False – When a dial command is issued, L-D Download will skip this unit.

`Public Property` DLTries() `As Integer`

- After L-D Download has read the exceedance and interval records, it will test to see if the records are in the proper format for entry into the SQL database. If one of the records is not, the program will download all of the records again. DLTries is the number of times the program should attempt to retrieve the E and I records before hanging up the line and declaring the download a failure.

- This property is not set by the user. To change this value, edit the number on line 1475 in Modem.vb and recompile the program.

`Public Property DialTries() As Integer`

- The number of times L-D Download will attempt to dial a unit before giving up and declaring the unit was unreachable.

`Public Property Initialize() As Boolean`

- True – When a connection with the unit has been established, the `ReconfigureUnit` method will execue. This method first resets the data on a unit, then sets the Q parameters to a default value specified in the code.
- False – When a connection with the unit has been established, the normal `Download` method will execute.

`Public Property SendCustP() As Boolean`

- True – Custom parameters will be sent to this unit. The custom parameters are stored in the variable `CustParams`.
- False – Custom parameters will not be sent to this unit.

<u>These properties contain identification data for a unit and are displayed in Unit Options and Unit Viewer.</u>

`Public Property UnitSerial() As String`

The serial number of the unit, read from the unit itself.

`Public Property UnitNum() As String`

The unit's L number, stored as "L00" through "L99." This is the main unique identifier of a LDUnit object in L-D Download. When the program needs to search for a unit, or sort the list of units, it uses this property as the search criterion.

`Public Property LockCode() As String`

The lock code, or password, needed to log onto the unit.

```
Public Property UnitLocation() As String
```

A text string detailing the physical location of the unit. This is what the Aberdeen staff uses to identify a unit.

```
Public Property UnitPhoneNum() As String
```

The telephone number of the unit. When the program wants to dial a unit, it calls this number.

```
Public Property UnitOwner() As Owner
```

The agency that operates the unit. This property is used most often by the program to let the scheduler or user set parameters that are common to just the Aberdeen units, or just the CERL units. For example, the Aberdeen units call in during the day, while the CERL ones do not; the CERL units may have a higher exceedance threshold than the Aberdeen units, etc. The Owner object itself is an enumeration (enum) defined in the LDUnit class. Your choices of owner are:

```
        Public Enum Owner
            Aberdeen
            CERL
            Nobody
        End Enum
```

```
Public Property LastDL() As DateTime
```

The date and time of the last successful download.

```
Public Property CustParams() As String
```

The custom parameter text for a unit, set through Unit Options.

These properties contain measurement and statistical data for a unit.

```
Public Property NumExceedances() As Integer
```

Number of exceedance records on the unit, as reported by the unit. Used to help verify the completeness of a download, and also displayed in Unit Options and Unit Viewer.

```
Public Property NumIntervals() As Integer
```

Number of interval records on the unit, as reported by the unit.

```
Public Property NumStartStops() As Integer
```

Number of log (L) records on the unit, as reported by the unit.

```
Public Property NumCalibrations() As Integer
```

Number of calibration (C) records on the unit, as reported by the unit.

```
Public Property BattVoltage() As Double
```

The unit's battery voltage.

```
Public Property ErrorString() As String
```

The last eight errors reported by the unit. In Unit Viewer, the program translates the numeric code reported by the unit into a verbose error report.

```
Public Property EList() As List(Of String)
Public Property IList() As List(Of String)
Public Property RList() As List(Of String)
Public Property QList() As List(Of String)
Public Property LList() As List(Of String)
Public Property CList() As List(Of String)
```

These lists store the exceedance, interval, read, query, log, and cal records, respectively, retrieved from the unit. When the program receives the data report, it breaks the report into lines. A line beginning with E is stored to a String in EList, I to IList, and so on. This structuring of the report data aids in checking the report for errors and storing the data to the SQL database or to a text file on disk. Once the report data are checked and written to disk, the data in these lists are deleted to free up memory and to avoid duplicate data being saved to disk.

```
Public Property ExcdThreshold() As Integer
```

The exceedance threshold (in decibels) read from the unit. This value is reported in Unit Options and Unit Viewer.

```
Public Property CalLevel() As Double
```

The calibration level (in decibels) read from the unit. This value is reported in Unit Options and Unit Viewer.

```
Public Property ExcdDay() As Integer
```

The daytime exceedance threshold set by the user.

```
Public Property ExcdNight() As Integer
```

The nighttime exceedance threshold set by the user. If the units are told to stop gathering data altogether at night, this value is not written to the unit.

```
Public Property TimerRun1() As String
```

The time of day at which the unit should start taking data.

```
Public Property TimerStop1() As String
```

The time of day at which the unit should stop taking data.

```
Public Property TimerChanged() As Boolean
```

- True – The user has changed TimerRun1 or TimerStop1 since the last download. L-D Download will write the new timer settings to the unit.
- False – The user has not changed TimerRun1 or TimerStop1 since the last download. L-D Download will not write new timer settings to the unit.

Appendix A: LD Download (Form1)

```vb
1 'Form1 is the main code it runs the main window.  Everything gets lauchned from here
2
3 Option Strict On Compiled code that this code uses
4
5 Imports System.Collections.Generic
6 'Generics - list or array where you don't need to know each index, you can sort through the array get the
      data.
7
8 Imports System.ComponentModel
9 'Allows you to use VB or .net components. (background worker type) class system
10
11 Imports System.Threading
12 Imports System.Data.SqlClient
13
14 Public Class Form1
15
16     'Local variables you need to iniliaze the memory space and for class variables the space is
      allocated and treated as a null value unit a value is assigned
17
18     Private Shared LDUnitList As New List(Of LDUnit)
19     Private WithEvents modem1, modem2, modem3, modem4 As Modem
20     Public Shared appset As AppSettings
21
22     'The String = "" initializes a variable and creates a memory address for that variable. This is good
      practice in coding.  If you don't initialize the declared variable then the code by crash.
23
24     Private Shared currLNum As String = ""  'used by GetUnit
25     Private Shared currLNum2 As String = "" 'used by UpdateStatusBoxHandler
26     Private wState As FormWindowState
27
28     'Enum is a type of data structure that vb internally uses e.g. 1 = add.  This also enables the code
      to give you the "n" number of choices when typing your code (e.g. add, create, etc.)
29
30     Private m_closeOK As Boolean
31     Private sBarNames As New List(Of String)
32     Private sBarPrecedingText As String = ""
33     Private sBarText As String = ""
34     Private sBarUCount As Integer = 0
35     Private stampedTime As TimeSpan
36     Private stampedTime2 As TimeSpan  'test!!
37     Private wasListening(3) As Boolean
38     Private Shared m_isUVopen As Boolean
39     Private Shared uv As New UnitView()
40     Private Shared m_isDaytime As Boolean
41     Private txtOut1h As New RichTextBox()
42     Private txtOut2h As New RichTextBox()
43     Private txtOut3h As New RichTextBox()
44     Private txtOut4h As New RichTextBox()
45     'txtoutXh are non-displayed text boxes which hold the text to be written to a log
46     'is a workaround for the slowdown that occurs when a RTB has to display a lot of text
47     Private rtfBody1 As New List(Of String)
48     Private rtfBody2 As New List(Of String)
49     Private rtfBody3 As New List(Of String)
50     Private rtfBody4 As New List(Of String)
51     Private header As String = "{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fcharset0
      Microsoft Sans Serif;}}" & vbLf & "{\colortbl ;\red0\green128\blue0;\red0\green0\blue0;\red255\green0\
      blue0;}" & vbLf & _
52   "\viewkind4\uc1\pard\f0\fs17"
53     Private footer As String = "\cf0\par" & vbLf & "}" & vbLf
54
55     Public Event SBUnitCounter As EventHandler(Of StatusBarEventArgs)
56     Public Event UpdateMessages As EventHandler(Of StatusBarEventArgs)
57
58
59     Public ReadOnly Property ReadLDUnitList() As List(Of LDUnit)
60         Get
61             Dim ulist As New List(Of LDUnit)
62             ulist.AddRange(LDUnitList)
63             Return ulist
64         End Get
65     End Property
66     Public Shared Property IsUVOpen() As Boolean
67         Get
68             Return m_isUVopen
69         End Get
70         Set(ByVal value As Boolean)
71             m_isUVopen = value
72         End Set
73     End Property
74     Public Shared Property IsDaytime() As Boolean
75         Get
76             Return m_isDaytime
77         End Get
78         Set(ByVal value As Boolean)
79             m_isDaytime = value
80         End Set
81     End Property
82     'IMPORTANT: To read from the list, then write back to the list,
83     'use the GetAllUnitsNoMark/ReturnAllUnits mechanism and let garbage collection handle the copied uList.
84     'Do NOT write a property - overwriting LDUnitList many times in succession led to lost and/or duplicate
85     'units
86
```

```vb
87
88      Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
89          Button1.Hide()
90          Button3.Hide()
91          Me.lblStatus.Text = ""
92          appset = New AppSettings()
93
94          Try
95              Using fs23 As New FileStream(appset.InstallPath & "app.dat", FileMode.Open)
96                  'New declares a new object or class
97                  'appset.InstallPath is C:\Program Files\LD_Download"
98
99                  Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
100                 appset = DirectCast(bf.Deserialize(fs23), AppSettings)
101
102             End Using
103         Catch ex As FileNotFoundException   'file does not exist - load default settings
104             appset = New AppSettings()
105         End Try
106
107         'routines to instantiate Modem and LDUnit objects
108         modem1 = New Modem(appset.Modem1Port, appset.Modem1Num, Modem.Mode.Standby)
109         modem2 = New Modem(appset.Modem2Port, appset.Modem2Num, Modem.Mode.Standby)
110         modem3 = New Modem(appset.Modem3Port, appset.Modem3Num, Modem.Mode.Standby)
111         modem4 = New Modem(appset.Modem4Port, appset.Modem4Num, Modem.Mode.Standby)
112
113         AddHandler modem1.UpdateOutBoxesEvent, AddressOf UpdateTxtOut1Handler
114         AddHandler modem2.UpdateOutBoxesEvent, AddressOf UpdateTxtOut2Handler
115         AddHandler modem3.UpdateOutBoxesEvent, AddressOf UpdateTxtOut3Handler
116         AddHandler modem4.UpdateOutBoxesEvent, AddressOf UpdateTxtOut4Handler
117
118         'Handlers will be removed automatically when application closes
119
120         Try
121
122             'If there is a problem with units being taken away from the list or more units being added
        to the list, the you can delete the units.dat and it will load in the defaults which are hard coded
        below
123
124             Using fs As New FileStream(appset.InstallPath & "units.dat", FileMode.Open)
125
126                 Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
127                 LDUnitList = DirectCast(bf.Deserialize(fs), List(Of LDUnit))
128
129             End Using
130         Catch ex As FileNotFoundException   'file does not exist
131             'Hard-code unit identifiers and phone numbers, for now
132             'units that are safe-ish to test
133             'LDUnitList.Add(New LDUnit("L38", "912176201877", LDUnit.Owner.CERL))
134             LDUnitList.Add(New LDUnit("L06", "14102737782", LDUnit.Owner.Aberdeen, "Perryman"))
135             LDUnitList.Add(New LDUnit("L07", "14106766443", LDUnit.Owner.Aberdeen, "Long Bar Harbor"))
136             LDUnitList.Add(New LDUnit("L08", "14107789023", LDUnit.Owner.Aberdeen, "Tolchester"))
137             LDUnitList.Add(New LDUnit("L13", "14109395258", LDUnit.Owner.Aberdeen, "Havre de Grace"))
138             LDUnitList.Add(New LDUnit("L15", "14103355733", LDUnit.Owner.Aberdeen, "Chase"))
139
140             'remaining APG units
141             LDUnitList.Add(New LDUnit("L01", "14107789024", LDUnit.Owner.Aberdeen, "Stoops Point"))
142             LDUnitList.Add(New LDUnit("L03", "14108100350", LDUnit.Owner.Aberdeen, "Howell Point"))
143             LDUnitList.Add(New LDUnit("L04", "14102751287", LDUnit.Owner.Aberdeen, "Grove Point"))
144             LDUnitList.Add(New LDUnit("L05", "14102721688", LDUnit.Owner.Aberdeen, "B Tower"))
145             LDUnitList.Add(New LDUnit("L09", "14107789025", LDUnit.Owner.Aberdeen, "Worton Point"))
146             LDUnitList.Add(New LDUnit("L10", "14102752384", LDUnit.Owner.Aberdeen, "Crystal Beach"))
147             LDUnitList.Add(New LDUnit("L11", "14103351960", LDUnit.Owner.Aberdeen, "Bowley's Quarter"))
148             LDUnitList.Add(New LDUnit("L12", "14103482211", LDUnit.Owner.Aberdeen, "Betterton"))
149             LDUnitList.Add(New LDUnit("L14", "14102737529", LDUnit.Owner.Aberdeen, "Aberdeen"))
150             LDUnitList.Add(New LDUnit("L16", "14102878283", LDUnit.Owner.Aberdeen, "Red Point"))
151             LDUnitList.Add(New LDUnit("L17", "14102878401", LDUnit.Owner.Aberdeen, "Charlestown"))
152             LDUnitList.Add(New LDUnit("L18", "14106761631", LDUnit.Owner.Aberdeen, "Edgewood"))
153             LDUnitList.Add(New LDUnit("L19", "14102870492", LDUnit.Owner.Aberdeen, "Turkey Point"))
154
155             'CERL units
156             LDUnitList.Add(New LDUnit("L21", "12175217254", LDUnit.Owner.CERL, "Sparrows Point"))
157             LDUnitList.Add(New LDUnit("L22", "12175218691", LDUnit.Owner.CERL, "Essex"))
158             LDUnitList.Add(New LDUnit("L23", "12176215594", LDUnit.Owner.CERL, "Baltimore"))
159             LDUnitList.Add(New LDUnit("L24", "12175218692", LDUnit.Owner.CERL, "Fork"))
160             LDUnitList.Add(New LDUnit("L25", "12175218693", LDUnit.Owner.CERL, "Bel Air"))
161             LDUnitList.Add(New LDUnit("L26", "12176201877", LDUnit.Owner.CERL, "Churchville"))
162             LDUnitList.Add(New LDUnit("L27", "12175218695", LDUnit.Owner.CERL, "Darlington"))
163             LDUnitList.Add(New LDUnit("L28", "12175218696", LDUnit.Owner.CERL, "Port Deposit"))
164             LDUnitList.Add(New LDUnit("L29", "12175218697", LDUnit.Owner.CERL, "Abingdon"))
165             LDUnitList.Add(New LDUnit("L30", "12175218698", LDUnit.Owner.CERL, "North East"))
166             LDUnitList.Add(New LDUnit("L31", "12175218800", LDUnit.Owner.CERL, "Elkton"))
167             LDUnitList.Add(New LDUnit("L32", "12175218802", LDUnit.Owner.CERL, "Chesapeake City"))
168             LDUnitList.Add(New LDUnit("L33", "12175218803", LDUnit.Owner.CERL, "Cecilton"))
169             LDUnitList.Add(New LDUnit("L34", "12175218807", LDUnit.Owner.CERL, "Millington"))
170             LDUnitList.Add(New LDUnit("L35", "12175218808", LDUnit.Owner.CERL, "Still Pond"))
171             LDUnitList.Add(New LDUnit("L36", "12175218809", LDUnit.Owner.CERL, "Chestertown"))
172             LDUnitList.Add(New LDUnit("L37", "12175218810", LDUnit.Owner.CERL, "Rock Hall"))
173
174         End Try
175
176         MakeUnitsDownloadable()
```

```vb
177
178         'The Me. accesses all the methods and variables available to you and allows you to select from    ↵
        those available.  It keeps you from having to type in the variable names over and over.  It also       ↵
        distiguishes between a class variable in that class and a local variable in that method.
179
180         'The Me. keyword more specifically denotes an instance class variable - that is, a new copy         ↵
        of this variable is created for each instance of a class in a program. If one instance of a class      ↵
        changes the instance variable "dollar", the other instances will not have their "dollar" variables     ↵
        changes as well.
181
182         'This is different for a 'Shared' variable, which is a class variable that is common to all          ↵
        instances of a class. If one instance of a class chages the data in a shared variable, the change will ↵
        be visible to all other instances of that class. Shared variables are not accessible using the Me      ↵
        keyword.
183
184         'other program settings
185         NotifyIcon1.Visible = False
186         Me.wState = Me.WindowState
187         Me.m_closeOK = False
188         Dim stampsec As Long = CLng(Date.Now.TimeOfDay.TotalSeconds)
189         Me.stampedTime = New TimeSpan(stampsec * 10000000)
190         'Time is stamped here so it can be used later in the process of determining when 25 minutes has      ↵
        elapsed.
191         IsUVOpen = False
192         IsDaytime = False
193
194         'test!!
195         Dim stamphr As Long = CLng(Date.Now.TimeOfDay.Hours)  'test - rounds time down to the nearest hour
196         Me.stampedTime2 = New TimeSpan(stamphr * 60 * 60 * 10000000)
197         Me.stampedTime2 = Me.stampedTime2.Subtract(New TimeSpan(0, 30, 0))
198         'end test!!
199
200         Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Clear)
201         RaiseEvent SBUnitCounter(Me, EA)
202         RaiseEvent UpdateMessages(Me, EA)
203         Me.lblStatus.Text = ""
204
205     End Sub
206     Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.  ↵
        Click
207
208         'Set this up as a test button.  It is not used in the program.
209
210         'test the date parsing function
211         Dim unitTime As String = "Wed 28Jun2006 13:10:15"
212         Dim hostdate As Date = Date.Now
213         Dim tOffset As Integer = 0
214         Dim tOffsetTemp As TimeSpan = New TimeSpan(0)
215         Dim tOffsetTemp2 As Double = -1
216         Dim unitdate As Date = New Date(1)
217         Dim hostDTTicks As Long = 0
218
219         If Not unitTime Like "#*" Then
220             unitTime = unitTime.Remove(0, 4)    'trim off the day of week from the date - it is redundant
221             '                                    and will introduce errors if it was set wrong on the 870
222         End If
223
224         unitdate = Date.Parse(unitTime)
225         unitdate = unitdate.AddMilliseconds(45)
226
227         For i As Integer = 0 To 7
228             hostdate = Date.Now
229             hostDTTicks += hostdate.Ticks >> 3
230             Threading.Thread.Sleep(125)
231         Next
232
233         Dim uuu As Date = New Date(hostDTTicks)
234
235         tOffsetTemp = hostdate.Subtract(unitdate)
236         tOffsetTemp2 = tOffsetTemp.TotalSeconds Mod 3600    'ignore the difference in hours -
237         '                                                     prevents time zone differences from
238         '                                                     screwing up the recorded time
239
240         tOffset = CInt(Math.Round(tOffsetTemp2, 0, MidpointRounding.AwayFromZero))
241
242         'MessageBox.Show(unitdate.ToString & vbCr & hostdate.ToString & vbCr & tOffsetTemp.ToString & vbCr & ↵
        tOffsetTemp2.ToString & vbCr & tOffset.ToString & vbCr & uuu.ToString, "Unit time, Host time, offset,  ↵
        offset seconds only")
243
244         MessageBox.Show(hostdate.Hour & ":" & hostdate.Minute & ":" & hostdate.Second)
245
246         'Dim ppp As Integer = Date.Now.DayOfWeek + 1
247         'MessageBox.Show(ppp.ToString)
248
249         'Dim conn As New SqlClient.SqlConnection("server=localhost; database=Complaint; Integrated Security= ↵
        SSPI")
250         'Dim cmd As New SqlCommand("", conn)
251
252         'Try
253         '    conn.Open()
254         '    cmd.CommandText = "SELECT count(UnitID) FROM Intervals"
255         '    Dim numIRows As Integer = CInt(cmd.ExecuteScalar())
```

```vb
256          '     For i As Integer = 1 To numIRows
257          '         cmd.CommandText = "Select Corrected_Time from Intervals where recordnum = " & i.ToString
258          '         Try
259          '             Dim dt3 As Date = DirectCast(cmd.ExecuteScalar, DateTime)
260          '             Dim dt4 As New TimeSpan(dt3.Ticks)
261          '             Dim hrs As Double = dt4.TotalHours
262          '             hrs = Math.Round(hrs, 2)
263          '             cmd.CommandText = "update Intervals set TotalHrs_Corrected = " & hrs.ToString & " where ⤡
        recordnum = " & i.ToString
264          '             cmd.ExecuteScalar()
265          '         Catch ex As Exception
266          '             'don't update the db
267          '         End Try
268          '         Application.DoEvents()
269          '     Next
270
271
272          '     cmd.CommandText = "SELECT count(UnitID) FROM EventData"
273          '     Dim numERows As Integer = CInt(cmd.ExecuteScalar())
274          '     For i As Integer = 1 To numERows
275          '         cmd.CommandText = "Select Corrected_Time from EventData where recordnum = " & i.ToString
276          '         Try
277          '             Dim dt3 As Date = DirectCast(cmd.ExecuteScalar, DateTime)
278          '             Dim dt4 As New TimeSpan(dt3.Ticks)
279          '             Dim hrs As Double = dt4.TotalHours
280          '             hrs = Math.Round(hrs, 2)
281          '             cmd.CommandText = "update EventData set TotalHrs_Corrected = " & hrs.ToString & " where ⤡
        recordnum = " & i.ToString
282          '             cmd.ExecuteScalar()
283          '         Catch ex As Exception
284          '             'don't update the db
285          '         End Try
286          '         Application.DoEvents()
287          '     Next
288
289          '     cmd.CommandText = "SELECT count(UnitID) FROM EventData"
290          '     numERows = CInt(cmd.ExecuteScalar())
291          '     For i As Integer = 1 To numERows
292          '         cmd.CommandText = "Select Date_Time from EventData where recordnum = " & i.ToString
293          '         Try
294          '             Dim dt3 As Date = DirectCast(cmd.ExecuteScalar, DateTime)
295          '             Dim dt4 As New TimeSpan(dt3.Ticks)
296          '             Dim hrs As Double = dt4.TotalHours
297          '             hrs = Math.Round(hrs, 2)
298          '             cmd.CommandText = "update EventData set TotalHrs = " & hrs.ToString & " where recordnum ⤡
        = " & i.ToString
299          '             cmd.ExecuteScalar()
300          '         Catch ex As Exception
301          '             'don't update the db
302          '         End Try
303          '         Application.DoEvents()
304          '     Next
305          'Catch ex As SqlException
306          '     MessageBox.Show(ex.Message)
307          'Finally
308          '     conn.Close()
309          'End Try
310
311
312      End Sub 'NOT USED IN THE PROGRAM
313
314
315      'The #Region is not used by the program, it is just a way to group a region of code together to make ⤡
        it easier to read
316
317
318  #Region "text box handlers"
319      'These methods run on separate threads, since they are event handlers,
320      'and therefore need delegates to point to methods which actually update the textboxes
321
322      'These Event handlers send text to the textboxes.
323
324      Public Sub UpdateTxtOut1Handler(ByVal sender As Object, ByVal e As UpdateOutBoxesEventArgs)
325          Dim deleg As UpdateTxtOutDelegate
326          Dim obj As Object() = {e}
327          deleg = New UpdateTxtOutDelegate(AddressOf UpdateTxtOut1)
328          txtOut1.Invoke(deleg, obj)
329      End Sub
330      Public Sub UpdateTxtOut2Handler(ByVal sender As Object, ByVal e As UpdateOutBoxesEventArgs)
331          Dim deleg As UpdateTxtOutDelegate
332          Dim obj As Object() = {e}
333          deleg = New UpdateTxtOutDelegate(AddressOf UpdateTxtOut2)
334          txtOut1.Invoke(deleg, obj)
335      End Sub
336      Public Sub UpdateTxtOut3Handler(ByVal sender As Object, ByVal e As UpdateOutBoxesEventArgs)
337          Dim deleg As UpdateTxtOutDelegate
338          Dim obj As Object() = {e}
339          deleg = New UpdateTxtOutDelegate(AddressOf UpdateTxtOut3)
340          txtOut3.Invoke(deleg, obj)
341      End Sub
342      Public Sub UpdateTxtOut4Handler(ByVal sender As Object, ByVal e As UpdateOutBoxesEventArgs)
343          Dim deleg As UpdateTxtOutDelegate
```

```vb
344            Dim obj As Object() = {e}
345            deleg = New UpdateTxtOutDelegate(AddressOf UpdateTxtOut4)
346            txtOut4.Invoke(deleg, obj)
347        End Sub
348
349    'Delegate declaration and worker methods for updating the text boxes
350    'These worker methods run on the UI thread
351    Public Delegate Sub UpdateTxtOutDelegate(ByVal e As UpdateOutBoxesEventArgs)
352    Public Sub UpdateTxtOut1(ByVal e As UpdateOutBoxesEventArgs)
353        Dim textColor As Color
354        If e.RWFlag = UpdateOutBoxesEventArgs.RW.Read Then
355            textColor = Color.Red
356        ElseIf e.RWFlag = UpdateOutBoxesEventArgs.RW.Write Then
357            textColor = Color.Black
358        Else
359            textColor = Color.Green
360        End If
361        With txtOut1h 'write a copy to the hidden box
362            .SelectionColor = textColor
363            .AppendText(e.AppendThisString)
364        End With
365
366        'ghetto fix
367        With txtOut1
368            .SelectionColor = textColor
369            .AppendText(e.AppendThisString)
370            .ScrollToCaret()
371        End With
372        If txtOut1.Lines.GetLength(0) > appset.MaxLines Then 'clear - yes, it's a ghetto fix
373            txtOut1.Clear()
374        End If
375        'end ghetto fix
376
377        ''generate RTF code
378        'Dim rtfStr As New System.Text.StringBuilder()
379        'Dim colorCtrl As String = ""
380
381        'Select Case e.RWFlag
382        '    Case UpdateOutBoxesEventArgs.RW.Notify : colorCtrl = vbLf & "\cf1 "
383        '    Case UpdateOutBoxesEventArgs.RW.Write : colorCtrl = vbLf & "\cf2 "
384        '    Case UpdateOutBoxesEventArgs.RW.Read : colorCtrl = vbLf & "\cf3 "
385        'End Select
386
387        'Dim echars() As Char = e.AppendThisString.ToCharArray()
388        'For Each c As Char In echars
389        '    If Microsoft.VisualBasic.AscW(c) > 127 Then
390        '        c = "_"c
391        '    End If
392        'Next
393        'e.AppendThisString = String.Concat(echars)
394        'e.AppendThisString = e.AppendThisString.Replace(vbLf, "\par ")
395        'e.AppendThisString = e.AppendThisString.Replace(vbCr, "\par ")
396
397        'Me.rtfBody1.Add(colorCtrl & e.AppendThisString)
398        'If Me.rtfBody1.Count > appset.MaxLines Then 'remove the oldest line(s)
399        '    Do
400        '        Me.rtfBody1.RemoveAt(0)
401        '    Loop Until Me.rtfBody1.Count <= appset.MaxLines
402        'End If
403
404        'For i As Integer = 0 To Me.rtfBody1.Count - 1
405        '    rtfStr.Append(Me.rtfBody1.Item(i))
406        'Next
407
408
409        ''post the rtf string to the screen
410        'txtOut1.SuspendLayout()
411        'With txtOut1
412        '    .Rtf = header & rtfStr.ToString & footer
413        '    .Select(Me.txtOut1.TextLength, 0)
414        '    If Not .Focused Then
415        '        .ScrollToCaret()
416        '        'if txtout1 has the focus, it will automatically scroll to the end of the selection
417        '    End If
418        'End With
419        'txtOut1.ResumeLayout()
420
421        '        'Me.Text = rtfBody1.Count.ToString & "   " & rtfBody2.Count.ToString & "   " & _
422        '        '  rtfBody3.Count.ToString & "   " & rtfBody4.Count.ToString
423    End Sub
424    Public Sub UpdateTxtOut2(ByVal e As UpdateOutBoxesEventArgs)
425        Dim textColor As Color
426        If e.RWFlag = UpdateOutBoxesEventArgs.RW.Read Then
427            textColor = Color.Red
428        ElseIf e.RWFlag = UpdateOutBoxesEventArgs.RW.Write Then
429            textColor = Color.Black
430        Else
431            textColor = Color.Green
432        End If
433        With txtOut2h 'write a copy to the hidden box
434            .SelectionColor = textColor
435            .AppendText(e.AppendThisString)
```

```vb
436          End With
437
438          'ghetto fix
439          With txtOut2
440              .SelectionColor = textColor
441              .AppendText(e.AppendThisString)
442              .ScrollToCaret()
443          End With
444          If txtOut2.Lines.GetLength(0) > appset.MaxLines Then 'clear - yes, it's a ghetto fix
445              txtOut2.Clear()
446          End If
447          'end ghetto fix
448
449          ''generate RTF code
450          'Dim rtfStr As New System.Text.StringBuilder()
451          'Dim colorCtrl As String = ""
452
453          'Select Case e.RWFlag
454          '    Case UpdateOutBoxesEventArgs.RW.Notify : colorCtrl = vbLf & "\cf1 "
455          '    Case UpdateOutBoxesEventArgs.RW.Write : colorCtrl = vbLf & "\cf2 "
456          '    Case UpdateOutBoxesEventArgs.RW.Read : colorCtrl = vbLf & "\cf3 "
457          'End Select
458
459          'Dim echars() As Char = e.AppendThisString.ToCharArray()
460          'For Each c As Char In echars
461          '    If Microsoft.VisualBasic.AscW(c) > 127 Then
462          '        c = "_"c
463          '    End If
464          'Next
465          'e.AppendThisString = String.Concat(echars)
466          'e.AppendThisString = e.AppendThisString.Replace(vbLf, "\par ")
467          'e.AppendThisString = e.AppendThisString.Replace(vbCr, "\par ")
468
469          'Me.rtfBody2.Add(colorCtrl & e.AppendThisString)
470          'If Me.rtfBody2.Count > appset.MaxLines Then 'remove the oldest line(s)
471          '    Do
472          '        Me.rtfBody2.RemoveAt(0)
473          '    Loop Until Me.rtfBody2.Count <= appset.MaxLines
474          'End If
475
476          'For i As Integer = 0 To Me.rtfBody2.Count - 1
477          '    rtfStr.Append(Me.rtfBody2.Item(i))
478          'Next
479
480
481          ''post the rtf string to the screen
482          'txtOut2.SuspendLayout()
483          'With txtOut2
484          '    .Rtf = header & rtfStr.ToString & footer
485          '    .Select(Me.txtOut2.TextLength, 0)
486          '    If Not .Focused Then
487          '        .ScrollToCaret()
488          '    End If
489          'End With
490          'txtOut2.ResumeLayout()
491
492          ''        Me.Text = rtfBody1.Count.ToString & "   " & rtfBody2.Count.ToString & "   " & _
493          ''        rtfBody3.Count.ToString & "   " & rtfBody4.Count.ToString
494      End Sub
495      Public Sub UpdateTxtOut3(ByVal e As UpdateOutBoxesEventArgs)
496          Dim textColor As Color
497          If e.RWFlag = UpdateOutBoxesEventArgs.RW.Read Then
498              textColor = Color.Red
499          ElseIf e.RWFlag = UpdateOutBoxesEventArgs.RW.Write Then
500              textColor = Color.Black
501          Else
502              textColor = Color.Green
503          End If
504          With txtOut3h 'write a copy to the hidden box
505              .SelectionColor = textColor
506              .AppendText(e.AppendThisString)
507          End With
508
509          'ghetto fix
510          With txtOut3
511              .SelectionColor = textColor
512              .AppendText(e.AppendThisString)
513              .ScrollToCaret()
514          End With
515          If txtOut3.Lines.GetLength(0) > appset.MaxLines Then 'clear - yes, it's a ghetto fix
516              txtOut3.Clear()
517          End If
518          'end ghetto fix
519
520
521          ''generate RTF code
522          'Dim rtfStr As New System.Text.StringBuilder()
523          'Dim colorCtrl As String = ""
524
525          'Select Case e.RWFlag
526          '    Case UpdateOutBoxesEventArgs.RW.Notify : colorCtrl = vbLf & "\cf1 "
527          '    Case UpdateOutBoxesEventArgs.RW.Write : colorCtrl = vbLf & "\cf2 "
```

```vb
528          '    Case UpdateOutBoxesEventArgs.RW.Read : colorCtrl = vbLf & "\cf3 "
529          'End Select
530
531          'Dim echars() As Char = e.AppendThisString.ToCharArray()
532          'For Each c As Char In echars
533          '    If Microsoft.VisualBasic.AscW(c) > 126 OrElse Microsoft.VisualBasic.AscW(c) < 9 OrElse _
534          '    Microsoft.VisualBasic.AscW(c) = 123 OrElse Microsoft.VisualBasic.AscW(c) = 125 OrElse _
535          '    Microsoft.VisualBasic.AscW(c) = 92 Then
536          '        c = "_"c
537          '    End If
538          'Next
539          'e.AppendThisString = String.Concat(echars)
540          'e.AppendThisString = e.AppendThisString.Replace(vbLf, "\par ")
541          'e.AppendThisString = e.AppendThisString.Replace(vbCr, "\par ")
542
543          'Me.rtfBody3.Add(colorCtrl & e.AppendThisString)
544          'If Me.rtfBody3.Count > appset.MaxLines Then 'remove the oldest line(s)
545          '    Do
546          '        Me.rtfBody3.RemoveAt(0)
547          '    Loop Until Me.rtfBody3.Count <= appset.MaxLines
548          'End If
549
550          'For i As Integer = 0 To Me.rtfBody3.Count - 1
551          '    rtfStr.Append(Me.rtfBody3.Item(i))
552          'Next
553
554          ''Me.rtbMessages.AppendText(Me.rtfBody3.Item(Me.rtfBody3.Count - 1))
555
556          ''post the rtf string to the screen
557          'txtOut3.SuspendLayout()
558          'With txtOut3
559          '    .Rtf = header & rtfStr.ToString & footer
560          '    .Select(Me.txtOut3.TextLength, 0)
561          '    If Not .Focused Then
562          '        .ScrollToCaret()
563          '    End If
564          'End With
565          'txtOut3.ResumeLayout()
566
567          '        'Me.Text = rtfBody1.Count.ToString & "   " & rtfBody2.Count.ToString & "   " & _
568          '        'rtfBody3.Count.ToString & "   " & rtfBody4.Count.ToString
569      End Sub
570      Public Sub UpdateTxtOut4(ByVal e As UpdateOutBoxesEventArgs)
571          Dim textColor As Color
572          If e.RWFlag = UpdateOutBoxesEventArgs.RW.Read Then
573              textColor = Color.Red
574          ElseIf e.RWFlag = UpdateOutBoxesEventArgs.RW.Write Then
575              textColor = Color.Black
576          Else
577              textColor = Color.Green
578          End If
579          With txtOut4h 'write a copy to the hidden box
580              .SelectionColor = textColor
581              .AppendText(e.AppendThisString)
582          End With
583
584          'ghetto fix
585          With txtOut4
586              .SelectionColor = textColor
587              .AppendText(e.AppendThisString)
588              .ScrollToCaret()
589          End With
590          If txtOut4.Lines.GetLength(0) > appset.MaxLines Then 'clear - yes, it's a ghetto fix
591              txtOut4.Clear()
592          End If
593          'end ghetto fix
594
595
596          ''generate RTF code
597          'Dim rtfStr As New System.Text.StringBuilder()
598          'Dim colorCtrl As String = ""
599
600          'Select Case e.RWFlag
601          '    Case UpdateOutBoxesEventArgs.RW.Notify : colorCtrl = vbLf & "\cf1 "
602          '    Case UpdateOutBoxesEventArgs.RW.Write : colorCtrl = vbLf & "\cf2 "
603          '    Case UpdateOutBoxesEventArgs.RW.Read : colorCtrl = vbLf & "\cf3 "
604          'End Select
605
606          'Dim echars() As Char = e.AppendThisString.ToCharArray()
607          'For Each c As Char In echars
608          '    If Microsoft.VisualBasic.AscW(c) > 127 Then
609          '        c = "_"c
610          '    End If
611          'Next
612          'e.AppendThisString = String.Concat(echars)
613          'e.AppendThisString = e.AppendThisString.Replace(vbLf, "\par ")
614          'e.AppendThisString = e.AppendThisString.Replace(vbCr, "\par ")
615
616          'Me.rtfBody4.Add(colorCtrl & e.AppendThisString)
617          'If Me.rtfBody4.Count > appset.MaxLines Then 'remove the oldest line(s)
618          '    Do
619          '        Me.rtfBody4.RemoveAt(0)
```

```vb
620     '    Loop Until Me.rtfBody4.Count <= appset.MaxLines
621     'End If
622
623     'For i As Integer = 0 To Me.rtfBody4.Count - 1
624     '    rtfStr.Append(Me.rtfBody4.Item(i))
625     'Next
626
627
628     ''post the rtf string to the screen
629     'txtOut4.SuspendLayout()
630     'With txtOut4
631     '    .Rtf = header & rtfStr.ToString & footer
632     '    .Select(Me.txtOut4.TextLength, 0)
633     '    If Not .Focused Then
634     '        .ScrollToCaret()
635     '    End If
636     'End With
637     'txtOut4.ResumeLayout()
638
639     '        'Me.Text = rtfBody1.Count.ToString & "    " & rtfBody2.Count.ToString & "   " & _
640     '        '    rtfBody3.Count.ToString & "   " & rtfBody4.Count.ToString
641     End Sub
642
643 #End Region
644
645 #Region "Status Bar Methods"
646
647     Public Sub SBUnitCounterHandler(ByVal sender As Object, ByVal e As StatusBarEventArgs) Handles Me. ↙
        SBUnitCounter, modem1.SBUnitCounter, modem2.SBUnitCounter, modem3.SBUnitCounter, modem4.SBUnitCounter
648
649         Dim labeltext As String = lblCounter.Text
650
651         Select Case e.Action
652             Case StatusBarEventArgs.SBAction.Populate
653                 'finds total # of units to be downloaded:
654                 Me.sBarUCount = 0
655                 For Each unit As LDUnit In LDUnitList
656                     If unit.IsDownloadDone = False AndAlso unit.IsDownloadInProgress = False Then
657                         Me.sBarUCount += 1
658                     End If
659                 Next
660                 labeltext = "Units left: " & Me.sBarUCount.ToString & "   "
661             Case StatusBarEventArgs.SBAction.Add
662                 Me.sBarUCount += 1
663                 labeltext = "Units left: " & Me.sBarUCount.ToString & "   "
664             Case StatusBarEventArgs.SBAction.Remove
665                 Me.sBarUCount -= 1
666                 If Me.sBarUCount < 0 Then
667                     Me.sBarUCount = 0
668                 End If
669                 labeltext = "Units left: " & Me.sBarUCount.ToString & "   "
670             Case StatusBarEventArgs.SBAction.Clear
671                 Me.sBarUCount = 0
672                 labeltext = ""
673             Case StatusBarEventArgs.SBAction.Cancelled
674             Case StatusBarEventArgs.SBAction.ChangePrecedingText
675             Case StatusBarEventArgs.SBAction.HideText
676                 labeltext = ""
677             Case StatusBarEventArgs.SBAction.ShowText
678                 labeltext = "Units left: " & Me.sBarUCount.ToString & "   "
679         End Select
680
681         lblCounter.Text = labeltext
682         uv.tsslUnitsLeft.Text = "    " & labeltext
683
684     End Sub
685
686     Private Sub Form1_UpdateMessages(ByVal sender As Object, ByVal e As StatusBarEventArgs) Handles Me. ↙
        UpdateMessages, modem1.UpdateMessages, modem2.UpdateMessages, modem3.UpdateMessages, modem4. ↙
        UpdateMessages
687         'updates the message box
688         'Populate mode writes "Dial sequence started at (date) (time)" and populates the (invisible) unit ↙
        list
689         'Add, Remove add and remove units
690         'ShowText runs when a dial sequence is finished and displays
691         ' "Dial sequence complete at (date) (time)"
692         ' "Units (units) did not download."
693
694         Dim unitString As New System.Text.StringBuilder
695         Dim trimchars() As Char = {"L"c, "0"c}
696
697         Select Case e.Action
698             Case StatusBarEventArgs.SBAction.Populate
699                 'clears the list, then populates it with units that are about to be downloaded
700                 Me.sBarNames.Clear()
701                 SyncLock LDUnitList
702                     For Each unit As LDUnit In LDUnitList
703                         If unit.IsDownloadDone = False AndAlso unit.IsDownloadInProgress = False Then
704                             Me.sBarNames.Add(unit.UnitNum & "  ")
705                         End If
706                     Next
707                 End SyncLock
```

```vb
708                    Me.sBarNames.Sort()
709                    For Each str As String In Me.sBarNames
710                        unitString.Append(str.TrimStart(trimchars))
711                    Next
712                    Me.rtbMessages.AppendText("Dial sequence started " & Date.Now.ToString("M/d/yyyy h:mm:ss tt" ⤹
     ) _
713                        & "." & vbLf)
714                    Me.rtbMessages.Select(Me.rtbMessages.TextLength, 0) 'moves caret to end of text box
715                    Me.rtbMessages.ScrollToCaret()
716                Case StatusBarEventArgs.SBAction.Add
717                    'add a unit name to SbarNames and display it
718                    Me.sBarNames.Add(e.UnitNum & "  ")
719                    Me.sBarNames.Sort()
720                Case StatusBarEventArgs.SBAction.Remove
721                    'remove a unit name and update the display
722                    currLNum2 = e.UnitNum
723                    Dim idx As Integer = Me.sBarNames.FindIndex(AddressOf IsUnitName)
724                    If idx > -1 Then
725                        Me.sBarNames.RemoveAt(idx)
726                    End If
727                    Me.sBarNames.Sort()
728                Case StatusBarEventArgs.SBAction.Clear
729                    Me.sBarNames.Clear()
730                    Me.sBarPrecedingText = ""
731                    Me.rtbMessages.Clear()
732                Case StatusBarEventArgs.SBAction.Cancelled
733                    Me.rtbMessages.AppendText("Dial sequence canceled " & Date.Now.ToString("M/d/yyyy h:mm:ss tt ⤹
     ") _
734                        & "." & vbLf)
735                    Me.rtbMessages.Select(Me.rtbMessages.TextLength, 0) 'moves caret to end of text box
736                    Me.rtbMessages.ScrollToCaret()
737                Case StatusBarEventArgs.SBAction.ChangePrecedingText
738                    'Add a line of text to the message box
739                    Me.rtbMessages.AppendText(e.PrecedingText)
740                Case StatusBarEventArgs.SBAction.HideText
741                Case StatusBarEventArgs.SBAction.ShowText
742                    'displays date and time download completes
743                    Me.sBarNames.Sort()
744                    For Each str As String In Me.sBarNames
745                        unitString.Append(str.TrimStart(trimchars))
746                    Next
747                    Me.rtbMessages.AppendText("Dial sequence completed " & Date.Now.ToString("M/d/yyyy h:mm:ss  ⤹
     tt") _
748                        & "." & vbLf)
749                    If Me.sBarNames.Count = 0 Then
750                        Me.rtbMessages.AppendText("All units downloaded successfully." & vbLf & vbLf)
751                    Else
752                        Me.rtbMessages.AppendText("Units " & unitString.ToString & "did not download." & vbLf &  ⤹
     vbLf)
753                        'uv.tsslUnitsLeft.Text = "Units " & unitString.ToString & "did not download."
754                    End If
755                    Me.rtbMessages.Select(Me.rtbMessages.TextLength, 0) 'moves caret to end of text box
756                    Me.rtbMessages.ScrollToCaret()
757
758            End Select
759        End Sub
760
761
762        Private Shared Function IsUnitName(ByVal str As String) As Boolean
763            If String.Compare(str, currLNum2 & "  ") = 0 Then
764                Return True
765            Else
766                Return False
767            End If
768        End Function
769
770
771 #End Region
772
773 #Region "Scheduler"
774
775        Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
776
777            Dim nightlyDL1 As New TimeSpan(21, 0, 0)    'use for test only
778            Dim nightlyDL2 As New TimeSpan(22, 15, 0)   'use for test only
779
780            Dim now As Date = Date.Now
781            Dim nowDay As DayOfWeek = now.DayOfWeek
782            Dim nowTime As TimeSpan = now.TimeOfDay
783            Dim diffTime As TimeSpan = nowTime.Subtract(Me.stampedTime)
784
785            'convert Now time to whole seconds
786            Dim nowtimesec As Long = CLng(nowTime.TotalSeconds)  'truncate the fractional seconds
787            nowTime = New TimeSpan(nowtimesec * 10000000)
788
789
790            'call the test unit at CERL once an hour at half past the hour
791            'If nowTime = New TimeSpan(0, 0, 0) Then 'subtract 24 hours - lets test run after midnight
792            '    stampedTime2 = stampedTime2.Subtract(New TimeSpan(24, 0, 0))
793            'End If
794
795            'Dim diffTime2 As TimeSpan = nowTime.Subtract(Me.stampedTime2)
```

```vb
796              'If diffTime2 > New TimeSpan(1, 0, 0) Then
797              '    'call unit 99 at 15 mins past the hour, every hour
798
799              '       SyncLock LDUnitList
800              '           For Each u As LDUnit In LDUnitList
801              '               If u.UnitNum = "L99" OrElse u.UnitNum = "L05" Then
802              '                   u.IsDownloadDone = False
803              '                   u.IsDownloadInProgress = False
804              '               Else
805              '                   u.IsDownloadDone = True
806              '               End If
807              '           Next
808              '       End SyncLock
809
810              '       CheckBox1.Checked = True
811              '       CheckBox2.Checked = True
812              '       CheckBox3.Checked = True
813              '       CheckBox4.Checked = False
814
815              '       If Not Me.BWDelayedClick.IsBusy Then
816              '           Me.BWDelayedClick.RunWorkerAsync()    'dial out
817              '       Else
818              '           Dim eaM As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingText)
819              '           eaM.PrecedingText = "Cannot start download at " & Date.Now.ToString("M/d/yyyy h:mm:ss tt") _
820              '               & "due to a busy worker thread." & vbLf & vbLf
821              '           RaiseEvent UpdateMessages(Me, eaM)
822              '       End If
823
824              '       Me.stampedTime2 = nowTime
825
826          'End If
827          'end test
828
829          Select Case nowDay
830              Case DayOfWeek.Sunday
831                  If appset.IsSunAWorkday Then
832                      If Form1.appset.WorkdayStart <= Form1.appset.WorkdayEnd Then
833                          If nowTime > Form1.appset.WorkdayStart AndAlso nowTime < Form1.appset.WorkdayEnd _
834                              IsDaytime = True
835                          Else
836                              IsDaytime = False
837                          End If
838                      Else 'ex. day starts at 6am and ends at 1am
839                          If nowTime > Form1.appset.WorkdayEnd AndAlso nowTime < Form1.appset.WorkdayStart _
840                              IsDaytime = False
841                          Else
842                              IsDaytime = True
843                          End If
844                      End If
845                      WorkdaySched(nowTime, diffTime)
846                  Else
847                      IsDaytime = False
848                      WeekendSched(nowTime)
849                  End If
850              Case DayOfWeek.Saturday
851                  If appset.IsSatAWorkday Then
852                      If Form1.appset.WorkdayStart <= Form1.appset.WorkdayEnd Then
853                          If nowTime > Form1.appset.WorkdayStart AndAlso nowTime < Form1.appset.WorkdayEnd _
854                              IsDaytime = True
855                          Else
856                              IsDaytime = False
857                          End If
858                      Else
859                          If nowTime > Form1.appset.WorkdayEnd AndAlso nowTime < Form1.appset.WorkdayStart _
860                              IsDaytime = False
861                          Else
862                              IsDaytime = True
863                          End If
864                      End If
865                      WorkdaySched(nowTime, diffTime)
866                  Else
867                      IsDaytime = False
868                      WeekendSched(nowTime)
869                  End If
870                  'otherwise
871              Case Else
872                  If Form1.appset.WorkdayStart <= Form1.appset.WorkdayEnd Then
873                      If nowTime > Form1.appset.WorkdayStart AndAlso nowTime < Form1.appset.WorkdayEnd Then
874                          IsDaytime = True
875                      Else
876                          IsDaytime = False
877                      End If
878                  Else 'ex. day starts at 6am and ends at 1am
879                      If nowTime > Form1.appset.WorkdayEnd AndAlso nowTime < Form1.appset.WorkdayStart Then
880                          IsDaytime = False
881                      Else
882                          IsDaytime = True
```

```vb
883                            End If
884                        End If
885                        WorkdaySched(nowTime, diffTime)
886
887            End Select
888
889        End Sub
890
891        Private Sub WorkdaySched(ByVal nowtime As TimeSpan, ByVal difftime As TimeSpan)
892            Dim e As New EventArgs
893
894            Select Case nowtime
895                Case appset.WeeknightDL '10:30pm by default
896                    'nightly download for all units - they should be stopped by timers by now
897                    Dim uList As List(Of LDUnit) = GetAllUnits()
898                    For Each unit As LDUnit In uList
899                        If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
900                            unit.AllowCallIns = True
901                            unit.ResetDataYN = True
902                            unit.ResetTimeYN = True 'uncomment when APG units are put on timer
903                            unit.DLTries = 0
904                            unit.IncludeLC = False
905                            unit.IncludeQ = False
906                            unit.IncludeR = False
907                            unit.IsDownloadDone = False
908                            unit.IsDownloadInProgress = False
909                        Else
910                            unit.AllowCallIns = False
911                            unit.ResetDataYN = True
912                            unit.ResetTimeYN = True
913                            unit.DLTries = 0
914                            unit.IncludeLC = False
915                            unit.IncludeQ = False
916                            unit.IncludeR = False
917                            unit.IsDownloadDone = False
918                            unit.IsDownloadInProgress = False
919                        End If
920                        'desired exceedance threshold is now determined in the Modem.Download method
921                    Next
922                    ReturnAllUnits(uList)
923                    CheckBox1.Checked = False
924                    CheckBox2.Checked = False
925                    CheckBox3.Checked = False  'no units should be calling in now
926                    CheckBox4.Checked = False
927
928                    If Not Me.BWDelayedClick.IsBusy Then
929                        Me.BWDelayedClick.RunWorkerAsync()     'dial out
930                    Else
931                        Dim eaM As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingText)
932                        eaM.PrecedingText = "Cannot start download at " & Date.Now.ToString("M/d/yyyy h:mm:ss tt↙
      ") _
933                            & "due to a busy worker thread." & vbLf & vbLf
934                        RaiseEvent UpdateMessages(Me, eaM)
935                    End If
936
937
938
939
940                    'Case appset.WorkdayStart '6am by default, soon to be obsolete
941                    '    'call APG units, enable dial-out mode
942                    '    'lower exceedance threshold to 100 dB for all units
943                    '    'Form1.IsDaytime = True
944
945                    '    Dim uList As List(Of LDUnit) = GetAllUnits()
946                    '    For Each unit As LDUnit In uList
947                    '        If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
948                    '            unit.AllowCallIns = True
949                    '            unit.ResetDataYN = True
950                    '            unit.IncludeQ = False
951                    '            unit.IncludeR = False
952                    '            unit.IsDownloadDone = False
953                    '            unit.IsDownloadInProgress = False
954                    '            'unit.ExcdThreshold = appset.WorkdayExcdThresh
955                    '        Else 'don't call
956                    '            unit.IsDownloadDone = True
957                    '            unit.AllowCallIns = False
958                    '            unit.ResetDataYN = True
959                    '            unit.IncludeQ = False
960                    '            unit.IncludeR = False
961                    '            'unit.ExcdThreshold = appset.WorkdayExcdThresh
962                    '        End If
963                    '    Next
964                    '    ReturnAllUnits(uList)
965
966                    '    'initialize the Listen timer
967                    '    stampedTime = nowtime
968
969                    '    'save the text from the night download 'is now done after every download
970                    '    'SaveFileDialog1.FileName = appset.InstallPath & "Logs\Log " & _
971                    '    'Date.Now.ToString("ddMMMyyyy HHmm") & ".rtf"
972
973                    '    'Dim ea2 As New CancelEventArgs(False)
```

```
974                       '    'SaveFileDialog1_FileOk(Me, ea2)
975
976                       '       CheckBox1.Checked = False
977                       '       CheckBox2.Checked = False
978                       '       CheckBox3.Checked = True      'leave line open to receive calls from units who could dial↙
     in
979                       '       CheckBox4.Checked = False
980
981                       '       If Not Me.BWDelayedClick.IsBusy Then
982                       '           Me.BWDelayedClick.RunWorkerAsync()     'dial out
983                       '       Else
984                       '           Dim eaM As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingText)
985                       '           eaM.PrecedingText = "Cannot start download at " & Date.Now.ToString("M/d/yyyy h:mm:↙
     ss tt") _
986                       '               & "due to a busy worker thread." & vbLf & vbLf
987                       '           RaiseEvent UpdateMessages(Me, eaM)
988                       '       End If
989
990
991                       'Case appset.WorkdayEnd '10pm by default, replace with Workday Night Download at 10pm
992                       '    '(so comment out this section and WorkdayStart)
993
994                       '       'call APG units, disable dial-out mode
995                       '       'Form1.IsDaytime = False
996
997                       '       Dim uList As List(Of LDUnit) = GetAllUnits()
998                       '       For Each unit As LDUnit In uList
999                       '           If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
1000                      '               unit.AllowCallIns = False 'change to True when APG units are put on the timer
1001                      '               unit.ResetDataYN = True
1002                      '               'unit.ResetTimeYN = True 'uncomment when APG units are put on timer
1003                      '               unit.DLTries = 0
1004                      '               unit.IncludeLC = False
1005                      '               unit.IncludeQ = False
1006                      '               unit.IncludeR = False
1007                      '               unit.IsDownloadDone = False
1008                      '               unit.IsDownloadInProgress = False
1009                      '               'unit.ExcdThreshold = appset.OtherExcdThresh
1010                      '           Else
1011                      '               unit.AllowCallIns = False
1012                      '               unit.ResetDataYN = True
1013                      '               unit.ResetTimeYN = True
1014                      '               unit.DLTries = 0
1015                      '               unit.IncludeLC = False
1016                      '               unit.IncludeQ = False
1017                      '               unit.IncludeR = False
1018                      '               unit.IsDownloadDone = False
1019                      '               unit.IsDownloadInProgress = False
1020                      '               'unit.ExcdThreshold = appset.OtherExcdThresh
1021                      '           End If
1022                      '       Next
1023                      '       ReturnAllUnits(uList)
1024                      '       CheckBox1.Checked = False
1025                      '       CheckBox2.Checked = False
1026                      '       CheckBox3.Checked = True      'this line is the first to receive calls
1027                      '       CheckBox4.Checked = False
1028
1029                      '       If Not Me.BWDelayedClick.IsBusy Then
1030                      '           Me.BWDelayedClick.RunWorkerAsync()     'dial out
1031                      '       Else
1032                      '           Dim eaM As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingText)
1033                      '           eaM.PrecedingText = "Cannot start download at " & Date.Now.ToString("M/d/yyyy h:mm:↙
     ss tt") _
1034                      '               & "due to a busy worker thread." & vbLf & vbLf
1035                      '           RaiseEvent UpdateMessages(Me, eaM)
1036                      '       End If
1037
1038
1039          End Select
1040
1041          're-check the Listen boxes
1042          If appset.DialInProg Then
1043              Me.stampedTime = nowtime
1044          End If
1045
1046          If (nowtime > appset.WorkdayStart.Add(New TimeSpan(0, 30, 0)) AndAlso nowtime < appset.WorkdayEnd _
1047          AndAlso (difftime > New TimeSpan(0, 25, 0)) AndAlso Not appset.DialInProg) Then
1048              If CheckBox1.Enabled Then
1049                  CheckBox1.Checked = True
1050              End If
1051              If CheckBox2.Enabled Then
1052                  CheckBox2.Checked = True
1053              End If
1054              If CheckBox3.Enabled Then
1055                  CheckBox3.Checked = True
1056              End If
1057              If CheckBox4.Enabled Then
1058                  CheckBox4.Checked = True
1059              End If
1060              Me.stampedTime = nowtime
1061          End If
1062
```

```vb
1063      End Sub
1064
1065      Private Sub WeekendSched(ByVal nowtime As TimeSpan)
1066          'Dim nightlyDL1 As New TimeSpan(21, 0, 0)    'use for test only
1067          Dim e As New EventArgs
1068
1069          Select Case nowtime
1070              Case appset.WeekendDL 'change this to 10pm
1071                  Dim uList As List(Of LDUnit) = GetAllUnits()
1072                  For Each unit As LDUnit In uList
1073                      unit.ResetDataYN = True
1074                      unit.ResetTimeYN = True
1075                      unit.DLTries = 0
1076                      unit.IncludeLC = False
1077                      unit.IncludeQ = True
1078                      unit.IncludeR = True
1079                      unit.IsDownloadDone = False
1080                      unit.IsDownloadInProgress = False
1081                  Next
1082                  ReturnAllUnits(uList)
1083
1084                  If Not Me.BWDelayedClick.IsBusy Then
1085                      Me.BWDelayedClick.RunWorkerAsync()     'dial out
1086                  Else
1087                      Dim eaM As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ChangePrecedingText)
1088                      eaM.PrecedingText = "Cannot start download at " & Date.Now.ToString("M/d/yyyy h:mm:ss tt↙
      ") _
1089                          & "due to a busy worker thread." & vbLf & vbLf
1090                      RaiseEvent UpdateMessages(Me, eaM)
1091                  End If
1092
1093          End Select
1094      End Sub
1095
1096  #End Region
1097
1098  #Region "Get/Return/Save Units"
1099      Public Shared Function GetUnit(ByVal LNum As String) As LDUnit
1100          'tries to return the unit specified by LNum
1101          'if LNum = -1, returns the next available unit
1102          'If Dial wants to get the next available unit, it needs to pass -2 as the unit number
1103          '   (this is an obsolete requirement - -1 and -2 do the same thing now)
1104          'Conversely, Listen will only be able to grab a specific unit if call-ins are enabled on that unit
1105
1106          Dim unitindex As Integer = 0
1107          Dim unit As LDUnit
1108          Dim unitToReturn As LDUnit 'use to ensure End Synclock is always called
1109
1110          SyncLock LDUnitList
1111              If String.Compare(LNum, "-1") = 0 OrElse String.Compare(LNum, "-2") = 0 Then
1112                  unitindex = LDUnitList.FindIndex(AddressOf NextUnit)
1113                  If unitindex > -1 Then
1114                      unit = LDUnitList.Item(unitindex)
1115                      LDUnitList.RemoveAt(unitindex)
1116                      unitToReturn = unit
1117                  Else    'no available units
1118                      unitToReturn = New LDUnit("null")
1119                  End If
1120              Else
1121                  currLNum = LNum
1122                  unitindex = LDUnitList.FindIndex(AddressOf SpecificUnit)
1123                  If unitindex > -1 Then
1124                      unit = LDUnitList.Item(unitindex)
1125                      LDUnitList.RemoveAt(unitindex)
1126                      unitToReturn = unit
1127                  Else
1128                      unitToReturn = New LDUnit("null")
1129                  End If
1130              End If
1131          End SyncLock
1132
1133          If Form1.IsUVOpen Then
1134              uv.RefreshToolStripMenuItem.PerformClick()
1135          End If
1136
1137          Return unitToReturn
1138
1139      End Function
1140
1141      Public Shared Sub ReturnUnit(ByVal unit As LDUnit)
1142          SyncLock LDUnitList
1143              unit.DLTries = 0
1144              LDUnitList.Add(unit)
1145          End SyncLock
1146
1147          If Form1.IsUVOpen Then
1148              uv.RefreshToolStripMenuItem.PerformClick()
1149          End If
1150      End Sub
1151
1152      Private Shared Function NextUnit(ByVal unit As LDUnit) As Boolean
1153          If unit.IsDownloadDone = False AndAlso unit.IsDownloadInProgress = False Then
```

```vb
1154                    Return True
1155            Else
1156                    Return False
1157            End If
1158        End Function
1159
1160        Private Shared Function SpecificUnit(ByVal unit As LDUnit) As Boolean
1161            If String.Compare(unit.UnitNum, currLNum) = 0 Then
1162                    Return True
1163            Else
1164                    Return False
1165            End If
1166        End Function
1167
1168        Private Shared Function NullUnit(ByVal unit As LDUnit) As Boolean
1169            If unit.UnitNum = "null" Then
1170                    Return True
1171            Else
1172                    Return False
1173            End If
1174        End Function
1175
1176        'Public Shared Function GetUnits() As List(Of LDUnit)
1177        '    'Gets only the units marked for download - used by Button2
1178
1179        '    Dim uList As New List(Of LDUnit)
1180        '    SyncLock LDUnitList
1181        '        Do
1182        '            uList.Add(GetUnit("-1"))
1183        '        Loop Until uList.Exists(AddressOf NullUnit)
1184        '    End SyncLock
1185        '    'get rid of the returned null unit
1186        '    uList.RemoveAll(AddressOf NullUnit)
1187
1188        '    Return uList
1189
1190        'End Function
1191
1192        Public Shared Function GetAllUnits() As List(Of LDUnit)
1193            Dim uList As New List(Of LDUnit)
1194
1195            MakeUnitsDownloadable()
1196            SyncLock LDUnitList
1197                uList.AddRange(LDUnitList)
1198                LDUnitList.Clear()
1199            End SyncLock
1200            'get rid of the returned null unit
1201            uList.RemoveAll(AddressOf NullUnit)
1202
1203            Return uList
1204
1205        End Function
1206
1207        Public Shared Function GetAllUnitsNoMark() As List(Of LDUnit)
1208            'gets all the units without changing their markings
1209            Dim ulist As New List(Of LDUnit)
1210
1211            SyncLock LDUnitList
1212                ulist.AddRange(LDUnitList)
1213                LDUnitList.Clear()
1214            End SyncLock
1215
1216            Return ulist
1217        End Function
1218
1219        Public Shared Sub ReturnAllUnits(ByRef uList As List(Of LDUnit))
1220            For Each unit As LDUnit In uList
1221                ReturnUnit(unit)
1222            Next
1223            'let garbage collection handle uList
1224        End Sub
1225
1226        Private Shared Sub MakeUnitsDownloadable()
1227            'run on Form1 thread?
1228
1229            SyncLock LDUnitList
1230                Dim remAtIndex As New List(Of Integer)
1231                For i As Integer = 0 To LDUnitList.Count - 1
1232                    LDUnitList.Item(i).IsDownloadDone = False
1233                    LDUnitList.Item(i).IsDownloadInProgress = False
1234                    LDUnitList.Item(i).DialTries = 0
1235                    If LDUnitList.Item(i).UnitNum = "null" OrElse LDUnitList.Item(i).UnitNum = "temp" Then
1236                        remAtIndex.Add(i)
1237                    End If
1238                Next
1239
1240                Dim res As String = ""
1241                For Each inte As Integer In remAtIndex
1242                    res &= inte.ToString & "  "
1243                Next
1244                'MessageBox.Show(res)
1245                For j As Integer = 0 To remAtIndex.Count - 1
```

```vb
1246                        LDUnitList.RemoveAt(remAtIndex.Item(j))
1247                Next
1248            End SyncLock
1249        End Sub
1250
1251        Private Sub SaveLDUnitList()
1252            Dim ucomp As New uCompare()
1253            Dim comp As Collections.Generic.IComparer(Of LDUnit) = ucomp
1254            SyncLock LDUnitList
1255                LDUnitList.Sort(comp)
1256                LDUnitList.TrimExcess()
1257                For Each unit As LDUnit In LDUnitList
1258                    With unit
1259                        .EList.Clear()
1260                        .IList.Clear()
1261                        .RList.Clear()
1262                        .QList.Clear()
1263                        .LList.Clear()
1264                        .CList.Clear()
1265                    End With
1266                Next
1267
1268                Using fs As New FileStream(appset.InstallPath & "units.dat", FileMode.Create)
1269
1270                    Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
1271                    bf.Serialize(fs, LDUnitList)
1272                End Using
1273            End SyncLock
1274        End Sub
1275
1276  #End Region
1277
1278  #Region "Dial/Listen Control"
1279
1280        Private Sub Button2_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2. ↵
        TextChanged
1281            'hide button if text = Dial - makes interface less confusing
1282            If Button2.Text = "Dial" Then
1283                Button2.Visible = False
1284                UnitOptionsToolStripMenuItem.Enabled = True
1285                Me.UnitOptionsToolStripMenuItem1.Enabled = True
1286            Else
1287                Button2.Visible = True
1288                UnitOptionsToolStripMenuItem.Enabled = False
1289                Me.UnitOptionsToolStripMenuItem1.Enabled = False
1290            End If
1291        End Sub
1292
1293        Public Delegate Sub PerformClickDelegate()
1294        Private Sub performclick()
1295            Dim e As New EventArgs()
1296            Button2_Click(Me, e)
1297        End Sub
1298        Private Sub BWDelayedClick_DoWork(ByVal sender As Object, ByVal e As System.ComponentModel. ↵
        DoWorkEventArgs) Handles BWDelayedClick.DoWork
1299            'performs a delayed button2 click without blocking the UI thread
1300            Dim pcdeleg As New PerformClickDelegate(AddressOf performclick)
1301            Dim normalDeleg As New ChangeCursor(AddressOf normalCursor)
1302            Dim waitDeleg As New ChangeCursor(AddressOf waitCursor)
1303
1304            Me.Invoke(waitDeleg)
1305            Thread.Sleep(5000)
1306            Me.Invoke(normalDeleg)
1307
1308            If Button2.Text <> "Dial" Then
1309                Button2.Invoke(pcdeleg)
1310                Me.Invoke(waitDeleg)
1311                Thread.Sleep(5000)
1312                Me.Invoke(normalDeleg)
1313                Button2.Invoke(pcdeleg)
1314            Else
1315                Button2.Invoke(pcdeleg)
1316            End If
1317        End Sub
1318
1319        Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2. ↵
        Click
1320            'Dial units using available modems
1321            'Insert code to determine which LDUnits to dial, and not to dial modems which are in Listen mode
1322            If Button2.Text = "Dial" Then
1323                appset.DialInProg = True 'a dial operation is in progress
1324
1325                If Not sender.Equals(Me) Then
1326                    MakeUnitsDownloadable()
1327                End If
1328
1329                'tell status bar how many units are left
1330                Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Populate)
1331                RaiseEvent SBUnitCounter(Me, EA)
1332                RaiseEvent UpdateMessages(Me, EA)
1333
1334                'run the background workers
```

```vb
1335                If modem1.modemMode = Modem.Mode.Standby AndAlso Not BW1.IsBusy Then
1336                    txtOut1.Clear()
1337                    txtOut1h.Clear()
1338                    rtfBody1.Clear()
1339                    modem1.modemMode = Modem.Mode.Dial
1340                    BW1.RunWorkerAsync()
1341                End If
1342                If modem2.modemMode = Modem.Mode.Standby AndAlso Not BW2.IsBusy Then
1343                    txtOut2.Clear()
1344                    txtOut2h.Clear()
1345                    rtfBody2.Clear()
1346                    modem2.modemMode = Modem.Mode.Dial
1347                    BW2.RunWorkerAsync()
1348                End If
1349                If modem3.modemMode = Modem.Mode.Standby AndAlso Not BW3.IsBusy Then
1350                    txtOut3.Clear()
1351                    txtOut3h.Clear()
1352                    rtfBody3.Clear()
1353                    modem3.modemMode = Modem.Mode.Dial
1354                    BW3.RunWorkerAsync()
1355                End If
1356                If modem4.modemMode = Modem.Mode.Standby AndAlso Not BW4.IsBusy Then
1357                    txtOut4.Clear()
1358                    txtOut4h.Clear()
1359                    rtfBody4.Clear()
1360                    modem4.modemMode = Modem.Mode.Dial
1361                    BW4.RunWorkerAsync()
1362                End If
1363                Button2.Text = "Cancel Downloads"
1364
1365            ElseIf Button2.Text <> "Dial" Then
1366                appset.DialInProg = False
1367                Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Clear)
1368                RaiseEvent SBUnitCounter(Me, EA)
1369
1370                If modem1.modemMode = Modem.Mode.Dial Then
1371                    BW1.CancelAsync()
1372                End If
1373                If modem2.modemMode = Modem.Mode.Dial Then
1374                    BW2.CancelAsync()
1375                End If
1376                If modem3.modemMode = Modem.Mode.Dial Then
1377                    BW3.CancelAsync()
1378                End If
1379                If modem4.modemMode = Modem.Mode.Dial Then
1380                    BW4.CancelAsync()
1381                End If
1382                If modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen _
1383                AndAlso modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen Then
1384                    Button2.Text = "Dial"
1385                End If
1386                Button2.Enabled = False
1387            End If
1388
1389
1390        End Sub
1391
1392        Private Sub BW1_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) ↵
           Handles BW1.DoWork
1393            'this method does not run on the UI thread
1394            Dim worker As BackgroundWorker = DirectCast(sender, BackgroundWorker)
1395
1396            If modem1.modemMode = Modem.Mode.Dial Then
1397                modem1.Dial(worker, e)
1398            ElseIf modem1.modemMode = Modem.Mode.Listen Then
1399                modem1.Listen(worker, e)
1400            End If
1401
1402        End Sub
1403        Private Sub BW1_RunWorkerCompleted(ByVal sender As System.Object, ByVal e As System.ComponentModel. ↵
           RunWorkerCompletedEventArgs) Handles BW1.RunWorkerCompleted
1404            'this method runs on the UI thread
1405            'This can either be reached via a normal completion or a user cancel
1406
1407            If e.Cancelled Then 'process was canceled either by a Listen check or a Cancel Download click
1408                CheckBox1.Enabled = True
1409                If modem1.modemMode = Modem.Mode.Listen Then
1410                    BW1.RunWorkerAsync()
1411                ElseIf modem1.modemMode = Modem.Mode.ListenCancel Then
1412                    'the listen box for this modem was just unchecked
1413                    If appset.DialInProg Then
1414                        'a dial is in progress
1415                        modem1.modemMode = Modem.Mode.Dial
1416                        BW1.RunWorkerAsync()
1417                    Else
1418                        modem1.modemMode = Modem.Mode.Standby
1419                        'do nothing else
1420                    End If
1421                Else
1422                    'a dial cancel was ordered by the user
1423                    modem1.modemMode = Modem.Mode.Standby
1424                    If Not (modem2.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
```

```vb
1425                    OrElse modem4.modemMode = Modem.Mode.Dial) Then
1426                        Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Cancelled)
1427                        RaiseEvent UpdateMessages(Me, EA)
1428                    End If
1429                End If
1430
1431                If Not (modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen _
1432                AndAlso modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen) Then
1433                    'it is appropriate to re-enable the dial button and menu
1434                    Button2.Enabled = True
1435                    Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1436                End If
1437
1438            Else
1439                'the process exited normally (if modem is in Dial mode)
1440                If modem1.modemMode = Modem.Mode.Dial Then
1441                    modem1.modemMode = Modem.Mode.Standby
1442                End If
1443
1444                If modem1.modemMode = Modem.Mode.Standby AndAlso Not appset.DialInProg AndAlso Not _
1445                    (modem2.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
1446                    OrElse modem4.modemMode = Modem.Mode.Dial) Then
1447                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ShowText)
1448                    RaiseEvent UpdateMessages(Me, EA)
1449                    EA.Action = StatusBarEventArgs.SBAction.HideText
1450                    RaiseEvent SBUnitCounter(Me, EA)
1451
1452                    'log the info on the screen to a file
1453                    If appset.SaveLogAfterDL Then
1454                        SaveFileDialog1.FileName = appset.InstallPath & "Logs\Log " & _
1455                            Date.Now.ToString("ddMMMyyyy HHmm") & ".rtf"
1456                        Dim ea2 As New CancelEventArgs(False)
1457                        SaveFileDialog1_FileOk(Me, ea2)
1458                    End If
1459
1460                    'enable the Listen boxes during the day
1461                    ReEnableListen()
1462                End If
1463
1464            End If
1465
1466        'if all modems are out of Dial mode, change button2 text to Dial and re-enable the dial button and ↵
        menu
1467        If Not (modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1468            OrElse modem3.modemMode = Modem.Mode.Dial OrElse modem4.modemMode = Modem.Mode.Dial) Then
1469            'appset.DialInProg = False
1470            Button2.Text = "Dial"
1471            Button2.Enabled = True
1472            Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1473        End If
1474
1475    End Sub
1476    Private Sub BW2_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) ↵
        Handles BW2.DoWork
1477
1478        Dim worker As BackgroundWorker = DirectCast(sender, BackgroundWorker)
1479
1480        If modem2.modemMode = Modem.Mode.Dial Then
1481            modem2.Dial(worker, e)
1482        ElseIf modem2.modemMode = Modem.Mode.Listen Then
1483            modem2.Listen(worker, e)
1484        End If
1485
1486    End Sub
1487    Private Sub BW2_RunWorkerCompleted(ByVal sender As System.Object, ByVal e As System.ComponentModel. ↵
        RunWorkerCompletedEventArgs) Handles BW2.RunWorkerCompleted
1488        'this method runs on the UI thread
1489        'This can either be reached via a normal completion, an exception, or a user cancel
1490        If e.Cancelled Then 'process was canceled either by a Listen check or a Cancel Download click
1491            CheckBox2.Enabled = True
1492            If modem2.modemMode = Modem.Mode.Listen Then
1493                BW2.RunWorkerAsync()
1494            ElseIf modem2.modemMode = Modem.Mode.ListenCancel Then
1495                'the listen box for this modem was just unchecked
1496                If appset.DialInProg Then
1497                    'a dial is in progress
1498                    modem2.modemMode = Modem.Mode.Dial
1499                    BW2.RunWorkerAsync()
1500                Else
1501                    modem2.modemMode = Modem.Mode.Standby
1502                    'do nothing else
1503                End If
1504            Else
1505                'a dial cancel was ordered by the user
1506                modem2.modemMode = Modem.Mode.Standby
1507                If Not (modem1.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
1508                OrElse modem4.modemMode = Modem.Mode.Dial) Then
1509                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Cancelled)
1510                    RaiseEvent UpdateMessages(Me, EA)
1511                End If
1512            End If
1513
```

```vb
1514                If Not (modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen _
1515                AndAlso modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen) Then
1516                    'it is appropriate to re-enable the dial button and menu
1517                    Button2.Enabled = True
1518                    Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1519                End If
1520
1521            Else
1522                'the process exited normally (if modem is in Dial mode)
1523                If modem2.modemMode = Modem.Mode.Dial Then
1524                    modem2.modemMode = Modem.Mode.Standby
1525                End If
1526
1527                If modem2.modemMode = Modem.Mode.Standby AndAlso Not appset.DialInProg AndAlso Not _
1528                (modem1.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
1529                OrElse modem4.modemMode = Modem.Mode.Dial) Then 'all modems finished dialing
1530                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ShowText)
1531                    RaiseEvent UpdateMessages(Me, EA)
1532                    EA.Action = StatusBarEventArgs.SBAction.HideText
1533                    RaiseEvent SBUnitCounter(Me, EA)
1534
1535                    'log the info on the screen to a file
1536                    If appset.SaveLogAfterDL Then
1537                        SaveFileDialog1.FileName = appset.InstallPath & "Logs\Log " & _
1538                            Date.Now.ToString("ddMMMyyyy HHmm") & ".rtf"
1539                        Dim ea2 As New CancelEventArgs(False)
1540                        SaveFileDialog1_FileOk(Me, ea2)
1541                    End If
1542
1543                    'enable the Listen boxes during the day
1544                    ReEnableListen()
1545                End If
1546            End If
1547
1548        'if all modems are out of Dial mode, change button2 text to Dial and re-enable the dial button and ↵
            menu
1549        If Not (modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1550            OrElse modem3.modemMode = Modem.Mode.Dial OrElse modem4.modemMode = Modem.Mode.Dial) Then
1551            appset.DialInProg = False
1552            Button2.Text = "Dial"
1553            Button2.Enabled = True
1554            Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1555        End If
1556
1557    End Sub
1558    Private Sub BW3_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) ↵
        Handles BW3.DoWork
1559
1560        Dim worker As BackgroundWorker = DirectCast(sender, BackgroundWorker)
1561
1562        If modem3.modemMode = Modem.Mode.Dial Then
1563            modem3.Dial(worker, e)
1564        ElseIf modem3.modemMode = Modem.Mode.Listen Then
1565            modem3.Listen(worker, e)
1566        End If
1567
1568    End Sub
1569    Private Sub BW3_RunWorkerCompleted(ByVal sender As System.Object, ByVal e As System.ComponentModel. ↵
        RunWorkerCompletedEventArgs) Handles BW3.RunWorkerCompleted
1570        'this method runs on the UI thread
1571        'This can either be reached via a normal completion, an exception, or a user cancel
1572
1573        If e.Cancelled Then 'process was canceled either by a Listen check or a Cancel Download click
1574            CheckBox3.Enabled = True
1575            If modem3.modemMode = Modem.Mode.Listen Then
1576                BW3.RunWorkerAsync()
1577            ElseIf modem3.modemMode = Modem.Mode.ListenCancel Then
1578                'the listen box for this modem was just unchecked
1579                If appset.DialInProg Then
1580                    'a dial is in progress
1581                    modem3.modemMode = Modem.Mode.Dial
1582                    BW3.RunWorkerAsync()
1583                Else
1584                    modem3.modemMode = Modem.Mode.Standby
1585                    'do nothing else
1586                End If
1587            Else
1588                'a dial cancel was ordered by the user
1589                modem3.modemMode = Modem.Mode.Standby
1590                If Not (modem2.modemMode = Modem.Mode.Dial OrElse modem1.modemMode = Modem.Mode.Dial _
1591                OrElse modem4.modemMode = Modem.Mode.Dial) Then
1592                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Cancelled)
1593                    RaiseEvent UpdateMessages(Me, EA)
1594                End If
1595            End If
1596
1597            If Not (modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen _
1598            AndAlso modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen) Then
1599                'it is appropriate to re-enable the dial button and menu
1600                Button2.Enabled = True
1601                Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1602            End If
```

```vb
1603
1604            Else
1605                'the process exited normally (if modem is in Dial mode)
1606                If modem3.modemMode = Modem.Mode.Dial Then
1607                    modem3.modemMode = Modem.Mode.Standby
1608                End If
1609
1610                If modem3.modemMode = Modem.Mode.Standby AndAlso Not appset.DialInProg AndAlso Not _
1611                (modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1612                OrElse modem4.modemMode = Modem.Mode.Dial) Then 'dial sequence is done
1613                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ShowText)
1614                    RaiseEvent UpdateMessages(Me, EA)
1615                    EA.Action = StatusBarEventArgs.SBAction.HideText
1616                    RaiseEvent SBUnitCounter(Me, EA)
1617
1618                    'log the info on the screen to a file
1619                    If appset.SaveLogAfterDL Then
1620                        SaveFileDialog1.FileName = appset.InstallPath & "Logs\Log " & _
1621                            Date.Now.ToString("ddMMMyyyy HHmm") & ".rtf"
1622                        Dim ea2 As New CancelEventArgs(False)
1623                        SaveFileDialog1_FileOk(Me, ea2)
1624                    End If
1625
1626                    'enable the Listen boxes during the day
1627                    ReEnableListen()
1628                End If
1629            End If
1630
1631        'if all modems are out of Dial mode, change button2 text to Dial and re-enable the dial button and ↵
        menu
1632        If Not (modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1633            OrElse modem3.modemMode = Modem.Mode.Dial OrElse modem4.modemMode = Modem.Mode.Dial) Then
1634            appset.DialInProg = False
1635            Button2.Text = "Dial"
1636            Button2.Enabled = True
1637            Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1638        End If
1639    End Sub
1640    Private Sub BW4_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) ↵
        Handles BW4.DoWork
1641
1642        Dim worker As BackgroundWorker = DirectCast(sender, BackgroundWorker)
1643
1644        If modem4.modemMode = Modem.Mode.Dial Then
1645            modem4.Dial(worker, e)
1646        ElseIf modem4.modemMode = Modem.Mode.Listen Then
1647            modem4.Listen(worker, e)
1648        End If
1649
1650    End Sub
1651    Private Sub BW4_RunWorkerCompleted(ByVal sender As System.Object, ByVal e As System.ComponentModel. ↵
        RunWorkerCompletedEventArgs) Handles BW4.RunWorkerCompleted
1652        'this method runs on the UI thread
1653        'This can either be reached via a normal completion, an exception, or a user cancel
1654
1655        If e.Cancelled Then 'process was canceled either by a Listen check or a Cancel Download click
1656            CheckBox4.Enabled = True
1657            If modem4.modemMode = Modem.Mode.Listen Then
1658                BW4.RunWorkerAsync()
1659            ElseIf modem4.modemMode = Modem.Mode.ListenCancel Then
1660                'the listen box for this modem was just unchecked
1661                If appset.DialInProg Then
1662                    'a dial is in progress
1663                    modem4.modemMode = Modem.Mode.Dial
1664                    BW4.RunWorkerAsync()
1665                Else
1666                    modem4.modemMode = Modem.Mode.Standby
1667                    'do nothing else
1668                End If
1669            Else
1670                'a dial or listen cancel was ordered by the user
1671                modem4.modemMode = Modem.Mode.Standby
1672                If Not (modem2.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
1673                OrElse modem1.modemMode = Modem.Mode.Dial) Then
1674                    Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Cancelled)
1675                    RaiseEvent UpdateMessages(Me, EA)
1676                End If
1677            End If
1678
1679            If Not (modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen _
1680            AndAlso modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen) Then
1681                'it is appropriate to re-enable the dial button and menu
1682                Button2.Enabled = True
1683                Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1684            End If
1685
1686        Else
1687            'the process exited normally (if modem is in Dial mode)
1688            If modem4.modemMode = Modem.Mode.Dial Then
1689                modem4.modemMode = Modem.Mode.Standby
1690            End If
1691
```

```vb
1692                 If modem4.modemMode = Modem.Mode.Standby AndAlso Not appset.DialInProg AndAlso Not _
1693                 (modem2.modemMode = Modem.Mode.Dial OrElse modem3.modemMode = Modem.Mode.Dial _
1694                 OrElse modem1.modemMode = Modem.Mode.Dial) Then 'dial process has completed
1695                     Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.ShowText)
1696                     RaiseEvent UpdateMessages(Me, EA)
1697                     EA.Action = StatusBarEventArgs.SBAction.HideText
1698                     RaiseEvent SBUnitCounter(Me, EA)
1699
1700                     'log the info on the screen to a file
1701                     If appset.SaveLogAfterDL Then
1702                         SaveFileDialog1.FileName = appset.InstallPath & "Logs\Log " & _
1703                             Date.Now.ToString("ddMMMyyyy HHmm") & ".rtf"
1704                         Dim ea2 As New CancelEventArgs(False)
1705                         SaveFileDialog1_FileOk(Me, ea2)
1706                     End If
1707
1708                     'enable the Listen boxes during the day
1709                     ReEnableListen()
1710                 End If
1711             End If
1712
1713         'if all modems are out of Dial mode, change button2 text to Dial and re-enable the dial button and
        menu
1714         If Not (modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1715             OrElse modem3.modemMode = Modem.Mode.Dial OrElse modem4.modemMode = Modem.Mode.Dial) Then
1716             appset.DialInProg = False
1717             Button2.Text = "Dial"
1718             Button2.Enabled = True
1719             Me.DownloadOneUnitToolStripMenuItem.Enabled = True
1720         End If
1721
1722     End Sub
1723
1724     Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
         CheckBox1.CheckedChanged
1725
1726         If CheckBox1.Checked Then
1727             modem1.modemMode = Modem.Mode.Listen
1728             If BW1.IsBusy = False Then
1729                 BW1.RunWorkerAsync()
1730             Else
1731                 'attempt to cancel the dial operation, then Listen
1732                 CheckBox1.Enabled = False
1733                 BW1.CancelAsync()
1734             End If
1735             If modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen AndAlso _
1736             modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen Then
1737                 Button2.Enabled = False
1738             End If
1739         Else
1740             'listen was deactivated, BW2 should always be busy
1741             If BW1.IsBusy Then
1742                 CheckBox1.Enabled = False
1743                 modem1.modemMode = Modem.Mode.ListenCancel
1744                 BW1.CancelAsync()
1745             End If
1746
1747         End If
1748     End Sub
1749
1750     Private Sub CheckBox2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
         CheckBox2.CheckedChanged
1751         If CheckBox2.Checked Then
1752             modem2.modemMode = Modem.Mode.Listen
1753             If BW2.IsBusy = False Then
1754                 BW2.RunWorkerAsync()
1755             Else
1756                 'attempt to cancel the dial operation, then Listen
1757                 CheckBox2.Enabled = False
1758                 BW2.CancelAsync()
1759             End If
1760             If modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen AndAlso _
1761             modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen Then
1762                 Button2.Enabled = False
1763             End If
1764         Else
1765             'listen was deactivated, BW2 should always be busy
1766             If BW2.IsBusy Then
1767                 CheckBox2.Enabled = False
1768                 modem2.modemMode = Modem.Mode.ListenCancel
1769                 BW2.CancelAsync()
1770             End If
1771
1772         End If
1773
1774     End Sub
1775
1776     Private Sub CheckBox3_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
         CheckBox3.CheckedChanged
1777
1778         If CheckBox3.Checked Then
1779             modem3.modemMode = Modem.Mode.Listen
```

```vb
1780                   If BW3.IsBusy = False Then
1781                       BW3.RunWorkerAsync()
1782                   Else
1783                       'attempt to cancel the dial operation, then Listen
1784                       CheckBox3.Enabled = False
1785                       BW3.CancelAsync()
1786                   End If
1787                   If modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen AndAlso _
1788                   modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen Then
1789                       Button2.Enabled = False
1790                   End If
1791               Else
1792                   'listen was deactivated, BW2 should always be busy
1793                   If BW3.IsBusy Then
1794                       CheckBox3.Enabled = False
1795                       modem3.modemMode = Modem.Mode.ListenCancel
1796                       BW3.CancelAsync()
1797                   End If
1798
1799               End If
1800       End Sub
1801
1802       Private Sub CheckBox4_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ⤦
              CheckBox4.CheckedChanged
1803
1804
1805           If CheckBox4.Checked Then
1806               modem4.modemMode = Modem.Mode.Listen
1807               If BW4.IsBusy = False Then
1808                   BW4.RunWorkerAsync()
1809               Else
1810                   'attempt to cancel the dial operation, then Listen
1811                   CheckBox4.Enabled = False
1812                   BW4.CancelAsync()
1813               End If
1814               If modem1.modemMode = Modem.Mode.Listen AndAlso modem2.modemMode = Modem.Mode.Listen AndAlso _
1815               modem3.modemMode = Modem.Mode.Listen AndAlso modem4.modemMode = Modem.Mode.Listen Then
1816                   Button2.Enabled = False
1817               End If
1818           Else
1819               'listen was deactivated, BW2 should always be busy
1820               If BW4.IsBusy Then
1821                   CheckBox4.Enabled = False
1822                   modem4.modemMode = Modem.Mode.ListenCancel
1823                   BW4.CancelAsync()
1824               End If
1825
1826           End If
1827       End Sub
1828
1829
1830       Private Sub CheckBox5_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ⤦
              CheckBox5.CheckedChanged
1831           If CheckBox5.Checked Then
1832               modem1.speakerOn = True
1833           Else
1834               modem1.speakerOn = False
1835           End If
1836       End Sub
1837
1838       Private Sub CheckBox6_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ⤦
              CheckBox6.CheckedChanged
1839           If CheckBox6.Checked Then
1840               modem2.speakerOn = True
1841           Else
1842               modem2.speakerOn = False
1843           End If
1844       End Sub
1845
1846       Private Sub CheckBox7_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ⤦
              CheckBox7.CheckedChanged
1847           If CheckBox7.Checked Then
1848               modem3.speakerOn = True
1849           Else
1850               modem3.speakerOn = False
1851           End If
1852       End Sub
1853
1854       Private Sub CheckBox8_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ⤦
              CheckBox8.CheckedChanged
1855           If CheckBox8.Checked Then
1856               modem4.speakerOn = True
1857           Else
1858               modem4.speakerOn = False
1859           End If
1860       End Sub
1861
1862
1863       Private Sub ReEnableListen()
1864           'immediately re-enables listen during the work week
1865           Dim ttt As TimeSpan = Date.Now.TimeOfDay
1866           Dim ddd As DayOfWeek = Date.Now.DayOfWeek
```

```vb
1867            If ttt > appset.WorkdayStart AndAlso ttt < appset.WorkdayEnd Then
1868                Select Case ddd
1869                    Case DayOfWeek.Sunday
1870                        'do nothing
1871                    Case DayOfWeek.Saturday
1872                        If appset.IsSatAWorkday Then
1873                            If CheckBox1.Enabled Then
1874                                CheckBox1.Checked = True
1875                            End If
1876                            If CheckBox2.Enabled Then
1877                                CheckBox2.Checked = True
1878                            End If
1879                            If CheckBox3.Enabled Then
1880                                CheckBox3.Checked = True
1881                            End If
1882                            If CheckBox4.Enabled Then
1883                                CheckBox4.Checked = True
1884                            End If
1885                            Me.stampedTime = ttt
1886                        End If
1887                        'otherwise
1888                    Case Else
1889                        If CheckBox1.Enabled Then
1890                            CheckBox1.Checked = True
1891                        End If
1892                        If CheckBox2.Enabled Then
1893                            CheckBox2.Checked = True
1894                        End If
1895                        If CheckBox3.Enabled Then
1896                            CheckBox3.Checked = True
1897                        End If
1898                        If CheckBox4.Enabled Then
1899                            CheckBox4.Checked = True
1900                        End If
1901                        Me.stampedTime = ttt
1902                End Select
1903            End If
1904        End Sub
1905
1906 #End Region
1907
1908 #Region "Menu Strip Controls"
1909     Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ↵
          Handles ExitToolStripMenuItem.Click
1910
1911        Me.m_closeOK = True
1912        SaveLDUnitList()
1913        Application.Exit()
1914
1915     End Sub
1916
1917     Private Sub DownloadOneUnitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System. ↵
          EventArgs) Handles DownloadOneUnitToolStripMenuItem.Click
1918        Form1.uv.Hide()
1919        Me.CmsForTrayIcon.Enabled = False
1920        Dim one As New OneUnitDialog()
1921        Dim dRes As DialogResult = one.ShowDialog(Me)
1922        Me.CmsForTrayIcon.Enabled = True
1923
1924        Select Case dRes
1925            Case Windows.Forms.DialogResult.OK
1926                appset.DialInProg = True
1927                Dim EA As New StatusBarEventArgs(StatusBarEventArgs.SBAction.Populate)
1928                RaiseEvent SBUnitCounter(Me, EA)
1929                RaiseEvent UpdateMessages(Me, EA)
1930
1931                If modem1.modemMode = Modem.Mode.Standby AndAlso Not BW1.IsBusy Then
1932                    txtOut1.Clear()
1933                    txtOut1h.Clear()
1934                    rtfBody1.Clear()
1935                    modem1.modemMode = Modem.Mode.Dial
1936                    BW1.RunWorkerAsync()
1937                End If
1938                If modem2.modemMode = Modem.Mode.Standby AndAlso Not BW2.IsBusy Then
1939                    txtOut2.Clear()
1940                    txtOut2h.Clear()
1941                    rtfBody2.Clear()
1942                    modem2.modemMode = Modem.Mode.Dial
1943                    BW2.RunWorkerAsync()
1944                End If
1945                If modem3.modemMode = Modem.Mode.Standby AndAlso Not BW3.IsBusy Then
1946                    txtOut3.Clear()
1947                    txtOut3h.Clear()
1948                    rtfBody3.Clear()
1949                    modem3.modemMode = Modem.Mode.Dial
1950                    BW3.RunWorkerAsync()
1951                End If
1952                If modem4.modemMode = Modem.Mode.Standby AndAlso Not BW4.IsBusy Then
1953                    txtOut4.Clear()
1954                    txtOut4h.Clear()
1955                    rtfBody4.Clear()
1956                    modem4.modemMode = Modem.Mode.Dial
```

```vb
1957                        BW4.RunWorkerAsync()
1958                    End If
1959                    If modem1.modemMode = Modem.Mode.Dial OrElse modem2.modemMode = Modem.Mode.Dial _
1960                        OrElse modem3.modemMode = Modem.Mode.Dial OrElse modem4.modemMode = Modem.Mode.Dial Then
1961                        'a dial operation is in progress - display the Cancel Downloads button
1962                        appset.DialInProg = True
1963                        Button2.Text = "Cancel Downloads"
1964                        Button2_TextChanged(Me, e)
1965                    End If
1966
1967                Case Else
1968                    'do nothing
1969            End Select
1970
1971            If IsUVOpen Then
1972                Form1.uv.Show()
1973                Me.Focus()
1974            End If
1975        End Sub
1976
1977        Private Sub SaveToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↙
               Handles SaveToolStripMenuItem.Click
1978
1979            Form1.uv.Hide()
1980            SaveFileDialog1.InitialDirectory = "Desktop"
1981            SaveFileDialog1.FileName = "Log " & Date.Now.ToString("ddMMMyyyy HHmm")
1982            SaveFileDialog1.ShowDialog(Me)
1983
1984            If Form1.IsUVOpen Then
1985                Form1.uv.Show()
1986                Me.Focus()
1987            End If
1988        End Sub
1989        Private Sub SaveFileDialog1_FileOk(ByVal sender As System.Object, ByVal e As System.ComponentModel.  ↙
               CancelEventArgs) Handles SaveFileDialog1.FileOk
1990
1991            Me.Cursor = Cursors.WaitCursor
1992            Me.CmsForTrayIcon.Enabled = False
1993            Me.SaveToolStripMenuItem.Enabled = False
1994
1995            Dim textBoxSave As New Windows.Forms.RichTextBox
1996
1997            textBoxSave.Multiline = True
1998            textBoxSave.ForeColor = Color.Blue
1999            textBoxSave.AppendText("Log file created " & Date.Now.ToString("M/d/yyyy HH:mm:ss") & vbLf & vbLf)
2000
2001            textBoxSave.AppendText("Messages" & vbLf & vbLf)
2002            textBoxSave.Select(textBoxSave.TextLength, 0) 'sets caret at the end of the text
2003            textBoxSave.SelectedRtf = Me.rtbMessages.Rtf
2004            textBoxSave.Select(textBoxSave.TextLength, 0)
2005
2006            textBoxSave.AppendText(vbLf & vbLf & vbLf & "Modem 1" & vbLf & vbLf & vbLf)
2007            textBoxSave.Select(textBoxSave.TextLength, 0)
2008            textBoxSave.SelectedRtf = Me.txtOut1h.Rtf
2009            textBoxSave.Select(textBoxSave.TextLength, 0)
2010
2011            textBoxSave.AppendText(vbLf & vbLf & vbLf & "Modem 2" & vbLf & vbLf & vbLf)
2012            textBoxSave.Select(textBoxSave.TextLength, 0)
2013            textBoxSave.SelectedRtf = Me.txtOut2h.Rtf
2014            textBoxSave.Select(textBoxSave.TextLength, 0)
2015
2016            textBoxSave.AppendText(vbLf & vbLf & vbLf & "Modem 3" & vbLf & vbLf & vbLf)
2017            textBoxSave.Select(textBoxSave.TextLength, 0)
2018            textBoxSave.SelectedRtf = Me.txtOut3h.Rtf
2019            textBoxSave.Select(textBoxSave.TextLength, 0)
2020
2021            textBoxSave.AppendText(vbLf & vbLf & vbLf & "Modem 4" & vbLf & vbLf & vbLf)
2022            textBoxSave.Select(textBoxSave.TextLength, 0)
2023            textBoxSave.SelectedRtf = Me.txtOut4h.Rtf
2024            textBoxSave.Select(textBoxSave.TextLength, 0)
2025
2026            textBoxSave.AppendText(vbLf & vbLf & "*** End of log ***" & vbLf & vbLf & vbLf)
2027
2028            textBoxSave.SaveFile(SaveFileDialog1.FileName)
2029
2030            'clear the hidden textboxes
2031            Me.txtOut1h.Clear()
2032            Me.txtOut2h.Clear()
2033            Me.txtOut3h.Clear()
2034            Me.txtOut4h.Clear()
2035
2036            Me.Cursor = Cursors.Default
2037            Me.CmsForTrayIcon.Enabled = True
2038            Me.SaveToolStripMenuItem.Enabled = True
2039        End Sub
2040
2041        Private Sub ViewUnitsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
                Handles ViewUnitsToolStripMenuItem.Click
2042            'opens the unit viewer - this window shouldn't be owned by Form1
2043            IsUVOpen = True
2044            If uv.Created Then
2045                uv.Show()
```

```vb
2046                    If uv.WindowState = FormWindowState.Minimized Then
2047                        uv.WindowState = FormWindowState.Normal
2048                    End If
2049                    uv.Focus()
2050                Else
2051                    uv = New UnitView()
2052                    uv.Show()
2053                End If
2054
2055        End Sub
2056
2057        Private Sub UnitOptionsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
            EventArgs) Handles UnitOptionsToolStripMenuItem.Click
2058
2059            Dim wasAnyoneListening As Boolean = False
2060
2061            'cancel Listen operation,
2062            Me.wasListening(0) = CheckBox1.Checked
2063            Me.wasListening(1) = CheckBox2.Checked
2064            Me.wasListening(2) = CheckBox3.Checked
2065            Me.wasListening(3) = CheckBox4.Checked
2066
2067            CheckBox1.Checked = False
2068            CheckBox2.Checked = False
2069            CheckBox3.Checked = False
2070            CheckBox4.Checked = False
2071
2072            'If Not Me.BWDelayedUO.IsBusy Then
2073            '    BWDelayedUO.RunWorkerAsync()
2074            'End If
2075
2076            For Each l As Boolean In wasListening
2077                If l Then
2078                    wasAnyoneListening = True
2079                End If
2080            Next
2081            If wasAnyoneListening Then
2082                Do
2083                    Me.Cursor = Cursors.WaitCursor
2084                    Thread.Sleep(10)
2085                    Application.DoEvents()
2086                Loop Until modem1.modemMode = Modem.Mode.Standby AndAlso modem2.modemMode = Modem.Mode.Standby _
2087                    AndAlso modem3.modemMode = Modem.Mode.Standby AndAlso modem4.modemMode = Modem.Mode.Standby
2088                Me.Cursor = Cursors.Default
2089            End If
2090
2091            'launch UnitOptions dialog in the UI thread
2092            Form1.uv.Hide()
2093            Me.CmsForTrayIcon.Enabled = False
2094            Dim uo As New UnitOptions()
2095            Dim dRes As DialogResult = uo.ShowDialog(Me)    'won't return to main form until this dialog box is
            closed
2096            Me.CmsForTrayIcon.Enabled = True
2097
2098            If dRes = Windows.Forms.DialogResult.OK Then
2099                SaveLDUnitList()
2100            End If
2101
2102            're-launch Listen on the units that were listening
2103            Me.CheckBox1.Checked = Me.wasListening(0)
2104            Me.CheckBox2.Checked = Me.wasListening(1)
2105            Me.CheckBox3.Checked = Me.wasListening(2)
2106            Me.CheckBox4.Checked = Me.wasListening(3)
2107
2108            If IsUVOpen Then
2109                Form1.uv.Show()
2110                Me.Focus() 'returns focus to Form1
2111            End If
2112        End Sub
2113
2114        Private Sub ViewDatabaseToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
            EventArgs) Handles ViewDatabaseToolStripMenuItem.Click
2115
2116            Diagnostics.Process.Start("Database Viewer.exe")
2117
2118        End Sub
2119
2120        Private Sub OptionsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
            Handles OptionsToolStripMenuItem.Click
2121            Form1.uv.Hide()
2122            Dim ass As New AppSettingsDialog()
2123            Dim dres As DialogResult = ass.ShowDialog()
2124
2125            If dres = Windows.Forms.DialogResult.OK Then
2126                modem1.modemPhoneNum = appset.Modem1Num
2127                modem2.modemPhoneNum = appset.Modem2Num
2128                modem3.modemPhoneNum = appset.Modem3Num
2129                modem4.modemPhoneNum = appset.Modem4Num
2130            End If
2131
2132            If IsUVOpen Then
2133                Form1.uv.Show()
```

```vb
2134                Me.Focus()
2135            End If
2136        End Sub
2137
2138        Private Delegate Sub ChangeCursor()
2139        Private Sub normalCursor()
2140            Me.Cursor = Cursors.Default
2141        End Sub
2142        Private Sub waitCursor()
2143            Me.Cursor = Cursors.WaitCursor
2144        End Sub
2145
2146 #End Region
2147
2148 #Region "Tray Icon Controls"
2149
2150        Private Sub RestoreToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↵
            Handles RestoreToolStripMenuItem.Click
2151            Me.WindowState = wState
2152        End Sub
2153
2154        Private Sub ManualDialToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.  ↵
            EventArgs) Handles ManualDialToolStripMenuItem.Click
2155            DownloadOneUnitToolStripMenuItem_Click(Me, e)
2156        End Sub
2157
2158        Private Sub ExitLDToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↵
            Handles ExitLDToolStripMenuItem.Click
2159            ExitToolStripMenuItem_Click(Me, e)
2160        End Sub
2161
2162        Private Sub Form1_Resize(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Resize
2163            If Me.WindowState = FormWindowState.Minimized Then
2164                Me.NotifyIcon1.Visible = True
2165                Me.ShowInTaskbar = False
2166            Else
2167                Me.NotifyIcon1.Visible = False
2168                Me.ShowInTaskbar = True
2169                wState = Me.WindowState
2170            End If
2171        End Sub
2172
2173        Private Sub NotifyIcon1_MouseDoubleClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms. ↵
            MouseEventArgs) Handles NotifyIcon1.MouseDoubleClick
2174            Me.WindowState = wState
2175        End Sub
2176
2177
2178        Private Sub SaveTextAsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.  ↵
            EventArgs) Handles SaveTextAsToolStripMenuItem.Click
2179
2180            Me.SaveToolStripMenuItem_Click(Me, New EventArgs())
2181        End Sub
2182
2183        Private Sub ViewUnitsToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.  ↵
            EventArgs) Handles ViewUnitsToolStripMenuItem1.Click
2184
2185            Me.ViewUnitsToolStripMenuItem_Click(Me, New EventArgs())
2186        End Sub
2187
2188        Private Sub UnitOptionsToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.  ↵
            EventArgs) Handles UnitOptionsToolStripMenuItem1.Click
2189
2190            Me.UnitOptionsToolStripMenuItem_Click(Me, New EventArgs())
2191        End Sub
2192
2193 #End Region
2194
2195        Private Sub Form1_FormClosingEventHandler(ByVal sender As Object, ByVal e As Windows.Forms.  ↵
            FormClosingEventArgs) Handles MyBase.FormClosing
2196            'idea from vb-helper.com
2197
2198            If m_closeOK = False Then
2199                e.Cancel = True
2200                Me.WindowState = FormWindowState.Minimized
2201                Form1_Resize(Me, e)
2202            Else
2203                'SaveLDUnitList() moved to the Exit handler because this method is not called if Form1 is  ↵
            minimized
2204                'and the user exits the program
2205            End If
2206        End Sub
2207
2208
2209
2210 End Class
2211
2212 Public Class UpdateOutBoxesEventArgs
2213        Inherits EventArgs
2214
2215        Private m_appendThisString As String
2216        Private m_RWFlag As RW
```

```vb
2217
2218     Public Enum RW
2219         Read
2220         Write
2221         Notify
2222     End Enum
2223
2224     Public Property AppendThisString() As String
2225         Get
2226             Return m_appendThisString
2227         End Get
2228         Set(ByVal value As String)
2229             m_appendThisString = value
2230         End Set
2231     End Property
2232
2233     Public Property RWFlag() As RW
2234         Get
2235             Return m_RWFlag
2236         End Get
2237         Set(ByVal value As RW)
2238             m_RWFlag = value
2239         End Set
2240     End Property
2241
2242     Public Sub New(ByVal value As String, ByVal flag As RW)
2243         m_appendThisString = value
2244         RWFlag = flag
2245     End Sub
2246
2247 End Class
2248
2249 Public Class StatusBarEventArgs
2250     Inherits EventArgs
2251
2252     Private m_action As SBAction
2253     Private m_precedingText As String
2254     Private m_unitNum As String
2255
2256     Public Property Action() As SBAction
2257         Get
2258             Return m_action
2259         End Get
2260         Set(ByVal value As SBAction)
2261             m_action = value
2262         End Set
2263     End Property
2264     Public Property PrecedingText() As String
2265         Get
2266             Return m_precedingText
2267         End Get
2268         Set(ByVal value As String)
2269             m_precedingText = value
2270         End Set
2271     End Property
2272     Public Property UnitNum() As String
2273         Get
2274             Return m_unitNum
2275         End Get
2276         Set(ByVal value As String)
2277             m_unitNum = value
2278         End Set
2279     End Property
2280
2281     Public Enum SBAction
2282         Populate
2283         Add
2284         Remove
2285         Clear
2286         Cancelled
2287         ChangePrecedingText
2288         HideText
2289         ShowText
2290     End Enum
2291
2292     Public Sub New(ByVal act As SBAction)    'use to clear or show the status bar text
2293         Action = act
2294         UnitNum = ""
2295     End Sub
2296
2297     Public Sub New(ByVal unit As String, ByVal act As SBAction)
2298         UnitNum = unit
2299         Action = act
2300     End Sub
2301
2302
2303
2304 End Class
2305
```

```xml
 1 <?xml version="1.0" encoding="utf-8" ?>
 2 <configuration>
 3     <configSections>
 4     </configSections>
 5     <connectionStrings>
 6         <add name="WindowsApplication1.My.MySettings.ComplaintConnectionString"
 7             connectionString="Data Source=localhost;Initial Catalog=Complaint;Integrated Security=True"
 8             providerName="System.Data.SqlClient" />
 9     </connectionStrings>
10     <system.diagnostics>
11         <sources>
12             <!-- This section defines the logging configuration for My.Application.Log -->
13             <source name="DefaultSource" switchName="DefaultSwitch">
14                 <listeners>
15                     <add name="FileLog"/>
16                     <!-- Uncomment the below section to write to the Application Event Log -->
17                     <!--<add name="EventLog"/>-->
18                 </listeners>
19             </source>
20         </sources>
21         <switches>
22             <add name="DefaultSwitch" value="Information" />
23         </switches>
24         <sharedListeners>
25             <add name="FileLog"
26                 type="Microsoft.VisualBasic.Logging.FileLogTraceListener, Microsoft.VisualBasic, Version=8.0.↵
   0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a, processorArchitecture=MSIL"
27                 initializeData="FileLogWriter"/>
28             <!-- Uncomment the below section and replace APPLICATION_NAME with the name of your application to↵
    write to the Application Event Log -->
29             <!--<add name="EventLog" type="System.Diagnostics.EventLogTraceListener" initializeData=            ↵
   "APPLICATION_NAME"/> -->
30         </sharedListeners>
31     </system.diagnostics>
32 </configuration>
33
```

Appendix B: Application Settings

```vb
1   Imports System.Windows.Forms
2
3   Public Class AppSettingsDialog
4
5       Private DA As Integer
6       Private ints(11) As Integer
7       Private excd1 As Integer
8       Private excd2 As Integer
9       Private mxLines As Integer
10      Private closeOK As Boolean = True
11
12
13      Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK_Button↲
        .Click
14          Dim modem1PortExists As Boolean = False
15          Dim modem2PortExists As Boolean = False
16          Dim modem3PortExists As Boolean = False
17          Dim modem4PortExists As Boolean = False
18          Dim noDuplicatePorts As Boolean = True
19
20          For i As Integer = 0 To My.Computer.Ports.SerialPortNames.Count - 1
21              If Me.cbbPort1.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
22                  modem1PortExists = True
23                  If Me.cbbPort2.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
24                  Me.cbbPort3.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
25                  Me.cbbPort4.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
26                      noDuplicatePorts = False
27                  End If
28              End If
29              If Me.cbbPort2.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
30                  modem2PortExists = True
31                  If Me.cbbPort1.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
32                  Me.cbbPort3.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
33                  Me.cbbPort4.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
34                      noDuplicatePorts = False
35                  End If
36              End If
37              If Me.cbbPort3.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
38                  modem3PortExists = True
39                  If Me.cbbPort2.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
40                  Me.cbbPort1.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
41                  Me.cbbPort4.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
42                      noDuplicatePorts = False
43                  End If
44              End If
45              If Me.cbbPort4.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
46                  modem4PortExists = True
47                  If Me.cbbPort2.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
48                  Me.cbbPort3.Text = My.Computer.Ports.SerialPortNames.Item(i) OrElse _
49                  Me.cbbPort1.Text = My.Computer.Ports.SerialPortNames.Item(i) Then
50                      noDuplicatePorts = False
51                  End If
52              End If
53          Next
54
55          If noDuplicatePorts Then
56              If modem1PortExists Then
57                  Form1.appset.Modem1Port = Me.cbbPort1.Text
58              Else
59                  Me.closeOK = False
60              End If
61              If modem2PortExists Then
62                  Form1.appset.Modem2Port = Me.cbbPort2.Text
63              Else
64                  Me.closeOK = False
65              End If
66              If modem3PortExists Then
67                  Form1.appset.Modem3Port = Me.cbbPort3.Text
68              Else
69                  Me.closeOK = False
70              End If
71              If modem4PortExists Then
72                  Form1.appset.Modem4Port = Me.cbbPort4.Text
73              Else
74                  Me.closeOK = False
75              End If
76          Else
77              closeOK = False
78          End If
79          If Not closeOK Then
80              MessageBox.Show("The modem's serial port name is entered incorrectly." & vbLf & _
81              "Please select a port from the pull-down list.", "Warning", MessageBoxButtons.OK, _
82              MessageBoxIcon.Warning)
83          End If
84
85
86          With Form1.appset
87              .InstallPath = tbInst.Text
88              .DialAttempts = DA - 1
89              .Modem1Num = tbM1.Text
90              .Modem2Num = tbM2.Text
91              .Modem3Num = tbM3.Text
```

```vb
 92                     .Modem4Num = tbM4.Text
 93                     .MaxLines = mxLines
 94                     .WeeknightDL = New TimeSpan(ints(0), ints(1), ints(2))
 95                     .WeekendDL = New TimeSpan(ints(3), ints(4), ints(5))
 96                     .WorkdayStart = New TimeSpan(ints(6), ints(7), ints(8))
 97                     .WorkdayEnd = New TimeSpan(ints(9), ints(10), ints(11))
 98                     .IsSatAWorkday = cbSat.Checked
 99                     .IsSunAWorkday = cbSun.Checked
100                     .StopAtNight = Me.cbStopAtNight.Checked
101                     .SaveLogAfterDL = cbSaveLog.Checked
102                     .Conn9600 = Me.cb9600.Checked
103                     .WorkdayExcdThresh = excd1
104                     .OtherExcdThresh = excd2
105                     .Serialize()
106             End With
107
108             'set each unit to the new timer value, if needed
109             Dim ulist As List(Of LDUnit) = Form1.GetAllUnitsNoMark
110             For Each u As LDUnit In ulist
111                 If u.TimerRun1 <> ints(6).ToString("d2") & ":" & ints(7).ToString("d2") _
112                 OrElse u.TimerStop1 <> ints(9).ToString("d2") & ":" & ints(10).ToString("d2") Then
113                     'enter new start and stop values, tell Modem to update the unit
114                     u.TimerRun1 = ints(6).ToString("d2") & ":" & ints(7).ToString("d2")
115                     u.TimerStop1 = ints(9).ToString("d2") & ":" & ints(10).ToString("d2")
116                     u.TimerChanged = True
117                 End If
118             Next
119             Form1.ReturnAllUnits(ulist)
120
121             If Me.closeOK Then
122                 Me.DialogResult = System.Windows.Forms.DialogResult.OK
123                 Me.Close()
124             Else
125                 Me.AppSettingsDialog_Load(Me, New EventArgs)
126             End If
127
128     End Sub
129
130     Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
        Cancel_Button.Click
131         Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
132         Me.Close()
133     End Sub
134
135     Private Sub AppSettingsDialog_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
        MyBase.Load
136
137         Me.DA = Form1.appset.DialAttempts
138         DA += 1
139         Me.tbInst.Text = Form1.appset.InstallPath
140         Me.tbDial.Text = DA.ToString
141         Me.tbLines.Text = Form1.appset.MaxLines.ToString 'this automatically sets me.mxLabel = maxLines
142         Me.tbM1.Text = Form1.appset.Modem1Num
143         Me.tbM2.Text = Form1.appset.Modem2Num
144         Me.tbM3.Text = Form1.appset.Modem3Num
145         Me.tbM4.Text = Form1.appset.Modem4Num
146         Me.tb1.Text = Form1.appset.WeeknightDL.Hours.ToString("00")
147         Me.tb2.Text = Form1.appset.WeeknightDL.Minutes.ToString("00")
148         Me.tb3.Text = Form1.appset.WeeknightDL.Seconds.ToString("00")
149         Me.tb4.Text = Form1.appset.WeekendDL.Hours.ToString("00")
150         Me.tb5.Text = Form1.appset.WeekendDL.Minutes.ToString("00")
151         Me.tb6.Text = Form1.appset.WeekendDL.Seconds.ToString("00")
152         Me.tb7.Text = Form1.appset.WorkdayStart.Hours.ToString("00")
153         Me.tb8.Text = Form1.appset.WorkdayStart.Minutes.ToString("00")
154         Me.tb9.Text = Form1.appset.WorkdayStart.Seconds.ToString("00")
155         Me.tb10.Text = Form1.appset.WorkdayEnd.Hours.ToString("00")
156         Me.tb11.Text = Form1.appset.WorkdayEnd.Minutes.ToString("00")
157         Me.tb12.Text = Form1.appset.WorkdayEnd.Seconds.ToString("00")
158         Me.cbSat.Checked = Form1.appset.IsSatAWorkday
159         Me.cbSun.Checked = Form1.appset.IsSunAWorkday
160         Me.cbStopAtNight.Checked = Form1.appset.StopAtNight
161         Me.cbSaveLog.Checked = Form1.appset.SaveLogAfterDL
162         Me.cb9600.Checked = Form1.appset.Conn9600
163
164         ints(0) = Form1.appset.WeeknightDL.Hours
165         ints(1) = Form1.appset.WeeknightDL.Minutes
166         ints(2) = Form1.appset.WeeknightDL.Seconds
167         ints(3) = Form1.appset.WeekendDL.Hours
168         ints(4) = Form1.appset.WeekendDL.Minutes
169         ints(5) = Form1.appset.WeekendDL.Seconds
170         ints(6) = Form1.appset.WorkdayStart.Hours
171         ints(7) = Form1.appset.WorkdayStart.Minutes
172         ints(8) = Form1.appset.WorkdayStart.Seconds
173         ints(9) = Form1.appset.WorkdayEnd.Hours
174         ints(10) = Form1.appset.WorkdayEnd.Minutes
175         ints(11) = Form1.appset.WorkdayEnd.Seconds
176
177         Me.cbSameNum.Checked = True
178
179         'hide stuff to change excd threshold - it's been moved to Unit Options
180
181         'populate the port list combo box(es)
```

```vb
182            cbbPort1.Items.Clear()
183            For i As Integer = 0 To My.Computer.Ports.SerialPortNames.Count - 1
184                cbbPort1.Items.Add(My.Computer.Ports.SerialPortNames.Item(i))
185                cbbPort2.Items.Add(My.Computer.Ports.SerialPortNames(i))
186                cbbPort3.Items.Add(My.Computer.Ports.SerialPortNames(i))
187                cbbPort4.Items.Add(My.Computer.Ports.SerialPortNames(i))
188            Next
189            Me.cbbPort1.Text = Form1.appset.Modem1Port
190            Me.cbbPort2.Text = Form1.appset.Modem2Port
191            Me.cbbPort3.Text = Form1.appset.Modem3Port
192            Me.cbbPort4.Text = Form1.appset.Modem4Port
193
194            tt.SetToolTip(Me.cb9600, "Tells the local modems to connect at 9600bps." & vbLf & _
195        "Clear this checkmark to let modems connect at the modem's maximum speed.")
196            tt.SetToolTip(Me.tbLines, "Sets the number of text lines to display in" & vbLf & _
197            "each of the four main windows." & vbLf & "Decrease this number if the program is sluggish upon
        restore." _
198            & vbLf & vbLf & "Does not affect how much text is written to the log file.")
199            Me.closeOK = True
200        End Sub
201
202
203
204        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.
        Click
205            Dim dres As DialogResult = FolderBrowserDialog1.ShowDialog()
206            If dres = Windows.Forms.DialogResult.OK Then
207                tbInst.Text = FolderBrowserDialog1.SelectedPath
208            End If
209        End Sub
210
211        Private Sub cbSameNum_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        cbSameNum.CheckedChanged
212            If cbSameNum.Checked Then
213                tbM2.Enabled = False
214                tbM3.Enabled = False
215                tbM4.Enabled = False
216            Else
217                tbM2.Enabled = True
218                tbM3.Enabled = True
219                tbM4.Enabled = True
220            End If
221        End Sub
222
223        Private Sub tbM1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbM1.
        TextChanged
224            If cbSameNum.Checked Then
225                tbM2.Text = tbM1.Text
226                tbM3.Text = tbM1.Text
227                tbM4.Text = tbM1.Text
228            End If
229        End Sub
230
231
232        Private Sub tbDial_LostFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles tbDial.
        LostFocus
233            Dim tryp As Boolean = Integer.TryParse(tbDial.Text, DA)
234
235            If tryp = False OrElse DA < 1 Then
236                MessageBox.Show("The number of dial attempts must be an integer greater than zero.", "Warning",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
237                DA = Form1.appset.DialAttempts
238                DA += 1
239                tbDial.Text = DA.ToString
240                tbDial.Focus()
241            End If
242        End Sub
243
244 #Region "Schedule text boxes"
245        Private Sub tb1_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb1.Enter
246            tb1.SelectAll()
247        End Sub
248
249
250        Private Sub Tb1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb1.
        TextChanged
251            Dim res As Integer
252            If tb1.Text.Length >= 2 Then
253                Dim tryp As Boolean = Integer.TryParse(tb1.Text, res)
254                If tryp AndAlso res >= 0 AndAlso res < 24 Then
255                    ints(0) = res
256                    tb2.Focus()
257                Else
258                    tb1.Clear()
259                    tb1.Focus()
260                End If
261            End If
262        End Sub
263        Private Sub tb2_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb2.Enter
264            tb2.SelectAll()
265        End Sub
266        Private Sub tb2_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb2.
```

```vb
                TextChanged
267                 Dim res As Integer
268                 If tb2.Text.Length >= 2 Then
269                     Dim tryp As Boolean = Integer.TryParse(tb2.Text, res)
270                     If tryp AndAlso res >= 0 AndAlso res < 60 Then
271                         ints(1) = res
272                         tb3.Focus()
273                     Else
274                         tb2.Clear()
275                         tb2.Focus()
276                     End If
277                 End If
278
279             End Sub
280
281             Private Sub tb3_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb3.Enter
282                 tb3.SelectAll()
283             End Sub
284
285             Private Sub tb3_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb3.  ↵
                TextChanged
286                 Dim res As Integer
287                 If tb3.Text.Length >= 2 Then
288                     Dim tryp As Boolean = Integer.TryParse(tb3.Text, res)
289                     If tryp AndAlso res >= 0 AndAlso res < 60 Then
290                         ints(2) = res
291                         tb4.Focus()
292                     Else
293                         tb3.Clear()
294                         tb3.Focus()
295                     End If
296                 End If
297
298             End Sub
299
300
301             Private Sub tb4_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb4.Enter
302                 tb4.SelectAll()
303             End Sub
304
305
306             Private Sub tb5_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb5.Enter
307                 tb5.SelectAll()
308             End Sub
309
310             Private Sub tb6_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb6.Enter
311                 tb6.SelectAll()
312             End Sub
313
314             Private Sub tb7_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb7.Enter
315                 tb7.SelectAll()
316             End Sub
317
318             Private Sub tb8_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb8.Enter
319                 tb8.SelectAll()
320             End Sub
321
322             Private Sub tb9_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb9.Enter
323                 tb9.SelectAll()
324             End Sub
325
326             Private Sub tb10_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb10.Enter
327                 tb10.SelectAll()
328             End Sub
329
330             Private Sub tb11_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb11.Enter
331                 tb11.SelectAll()
332             End Sub
333
334             Private Sub tb12_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles tb12.Enter
335                 tb12.SelectAll()
336             End Sub
337
338             Private Sub tb4_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb4.  ↵
                TextChanged
339                 Dim res As Integer
340                 If tb4.Text.Length >= 2 Then
341                     Dim tryp As Boolean = Integer.TryParse(tb4.Text, res)
342                     If tryp AndAlso res >= 0 AndAlso res < 24 Then
343                         ints(3) = res
344                         tb5.Focus()
345                     Else
346                         tb4.Clear()
347                         tb4.Focus()
348                     End If
349                 End If
350             End Sub
351
352             Private Sub tb5_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb5.  ↵
                TextChanged
353                 Dim res As Integer
354                 If tb5.Text.Length >= 2 Then
```

```vb
355                    Dim tryp As Boolean = Integer.TryParse(tb5.Text, res)
356                    If tryp AndAlso res >= 0 AndAlso res < 60 Then
357                        ints(4) = res
358                        tb6.Focus()
359                    Else
360                        tb5.Clear()
361                        tb5.Focus()
362                    End If
363                End If
364        End Sub
365
366        Private Sub tb6_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb6.
           TextChanged
367            Dim res As Integer
368            If tb6.Text.Length >= 2 Then
369                Dim tryp As Boolean = Integer.TryParse(tb6.Text, res)
370                If tryp AndAlso res >= 0 AndAlso res < 60 Then
371                    ints(5) = res
372                    tb7.Focus()
373                Else
374                    tb6.Clear()
375                    tb6.Focus()
376                End If
377            End If
378        End Sub
379
380        Private Sub tb7_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb7.
           TextChanged
381            Dim res As Integer
382            If tb7.Text.Length >= 2 Then
383                Dim tryp As Boolean = Integer.TryParse(tb7.Text, res)
384                If tryp AndAlso res >= 0 AndAlso res < 24 Then
385                    ints(6) = res
386                    tb8.Focus()
387                Else
388                    tb7.Clear()
389                    tb7.Focus()
390                End If
391            End If
392        End Sub
393
394        Private Sub tb8_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb8.
           TextChanged
395            Dim res As Integer
396            If tb8.Text.Length >= 2 Then
397                Dim tryp As Boolean = Integer.TryParse(tb8.Text, res)
398                If tryp AndAlso res >= 0 AndAlso res < 60 Then
399                    ints(7) = res
400                    tb9.Focus()
401                Else
402                    tb8.Clear()
403                    tb8.Focus()
404                End If
405            End If
406
407        End Sub
408
409        Private Sub tb9_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb9.
           TextChanged
410            Dim res As Integer
411            If tb9.Text.Length >= 2 Then
412                Dim tryp As Boolean = Integer.TryParse(tb9.Text, res)
413                If tryp AndAlso res >= 0 AndAlso res < 60 Then
414                    ints(8) = res
415                    tb10.Focus()
416                Else
417                    tb9.Clear()
418                    tb9.Focus()
419                End If
420            End If
421        End Sub
422
423        Private Sub tb10_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb10.
           TextChanged
424            Dim res As Integer
425            If tb10.Text.Length >= 2 Then
426                Dim tryp As Boolean = Integer.TryParse(tb10.Text, res)
427                If tryp AndAlso res >= 0 AndAlso res < 24 Then
428                    ints(9) = res
429                    tb11.Focus()
430                Else
431                    tb10.Clear()
432                    tb10.Focus()
433                End If
434            End If
435
436        End Sub
437
438        Private Sub tb11_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb11.
           TextChanged
439            Dim res As Integer
440            If tb11.Text.Length >= 2 Then
```
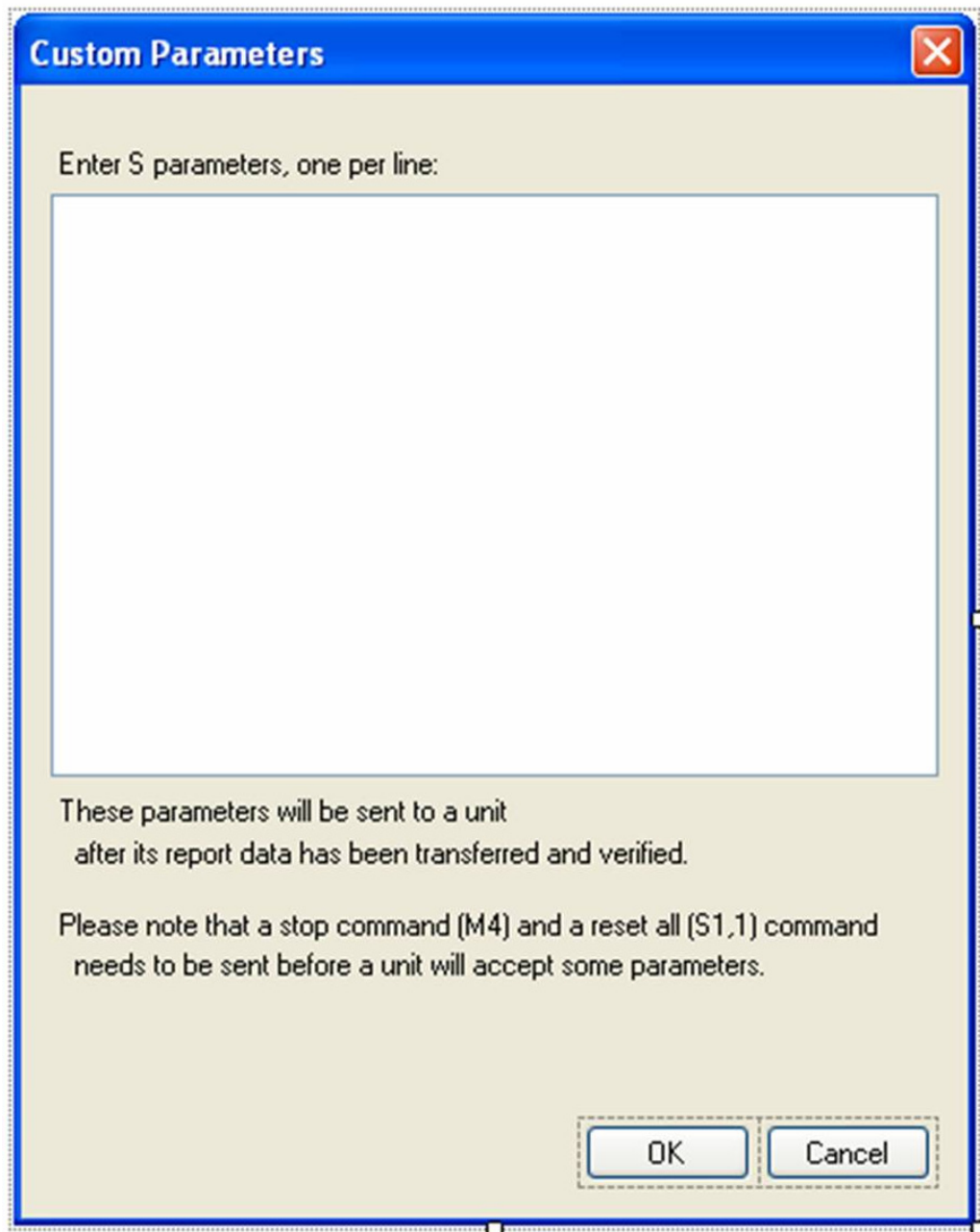
```vb
441                Dim tryp As Boolean = Integer.TryParse(tb11.Text, res)
442                If tryp AndAlso res >= 0 AndAlso res < 60 Then
443                    ints(10) = res
444                    tb12.Focus()
445                Else
446                    tb11.Clear()
447                    tb11.Focus()
448                End If
449            End If
450        End Sub
451
452        Private Sub tb12_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tb12.
           TextChanged
453            Dim res As Integer
454            If tb12.Text.Length >= 2 Then
455                Dim tryp As Boolean = Integer.TryParse(tb12.Text, res)
456                If tryp AndAlso res >= 0 AndAlso res < 60 Then
457                    ints(11) = res
458                    tb12.Focus()
459                Else
460                    tb12.Clear()
461                    tb12.Focus()
462                End If
463            End If
464        End Sub
465
466 #End Region
467
468        'Private Sub tbExcd1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
469        '    'makes sure the text entered is a number between 0 and 1000
470        '    Dim res As Integer
471        '    Dim tryp As Boolean = Integer.TryParse(tbExcd1.Text, res)
472
473        '    If tryp = True AndAlso res > -1 AndAlso res < 1000 Then
474        '        Me.excd1 = res
475        '    End If
476        'End Sub
477
478        'Private Sub tbExcd2_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
479        '    'makes sure the text entered is a number between 0 and 1000
480        '    Dim res As Integer
481        '    Dim tryp As Boolean = Integer.TryParse(tbExcd2.Text, res)
482
483        '    If tryp = True AndAlso res > -1 AndAlso res < 1000 Then
484        '        Me.excd2 = res
485        '    End If
486        'End Sub
487
488        Private Sub tbLines_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
           tbLines.TextChanged
489            'makes sure the text entered is greater than zero
490            Dim res As Integer
491            Dim tryp As Boolean = Integer.TryParse(tbLines.Text, res)
492
493            If tryp = True AndAlso res > 19 Then
494                Me.mxLines = res
495            End If
496        End Sub
497
498
499 End Class
500
501
```

Appendix C: Custom Parameters

Custom Parameters

Enter S parameters, one per line:

These parameters will be sent to a unit
after its report data has been transferred and verified.

Please note that a stop command (M4) and a reset all (S1,1) command
needs to be sent before a unit will accept some parameters.

[OK]　[Cancel]

```vb
1  Imports System.Windows.Forms
2
3  Public Class CustomParamsDialog
4
5      Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.
       Click
6          Me.DialogResult = System.Windows.Forms.DialogResult.OK
7          Me.Close()
8      End Sub
9
10     Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
       Cancel_Button.Click
11         Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
12         Me.Close()
13     End Sub
14
15     Private Sub CustomParamsDialog_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
       MyBase.Load
16         Me.TextBox1.Focus()
17
18     End Sub
19 End Class
20
```

```vb
1  'LDUnit
2
3  'an object which represents one LD unit
4
5  Imports System
6  Imports System.IO
7  Imports System.IO.Ports
8  'Imports System.Text
9  Imports System.Threading
10 Imports System.Data.SqlClient
11
12 <Serializable()> _
13 Public Class LDUnit
14     'path to which to serialize objects
15     Protected m_installPath As String
16
17     'report printing parameters
18     'The UI and scheduler will set these properties:
19
20     Protected m_includeR As Boolean
21     Protected m_includeLC As Boolean
22     Protected m_includeQ As Boolean
23     Protected m_allowCallIns As Boolean
24     Protected m_resetDataYN As Boolean
25     Protected m_resetTimeYN As Boolean
26     Protected m_isDownloadDone As Boolean
27     Protected m_isDownloadInProgress As Boolean
28     Protected m_isEnabled As Boolean
29     Protected m_dlTries As Integer
30     Protected m_dialTries As Integer
31     Protected m_initialize As Boolean
32     Protected m_sendCustP As Boolean
33
34     '   Unit serial number
35     '   Unit symbolic "L" number
36     '   Unit password (11111111 or 22222222)
37     'unit location (helps in identifying it)
38     'unit phone number
39
40     Protected m_unitSerial As String = ""
41     Protected m_unitLNum As String = ""
42     Protected m_lockCode As String = ""
43     Protected m_unitLocation As String = ""
44     Protected m_unitPhoneNum As String = ""
45     Protected m_unitOwner As Owner
46     Protected m_lastDL As DateTime
47     Protected m_custParams As String = ""
48
49     'These are R and Q parameters read from the unit upon download
50     Protected m_numExceedances As Integer = -1
51     Protected m_numIntervals As Integer = -1
52     Protected m_numStartStops As Integer = -1
53     Protected m_numCalibrations As Integer = -1
54     Protected m_battVoltage As Double = -1.1
55     Protected m_errorString As String = ""
56     Protected m_eList As New List(Of String)
57     Protected m_iList As New List(Of String)
58     Protected m_rList As New List(Of String)
59     Protected m_qList As New List(Of String)
60     Protected m_lList As New List(Of String)
61     Protected m_cList As New List(Of String)
62     Protected m_excdThreshold As Integer = -1
63     Protected m_calLevel As Double = -1
64     Protected m_excdThreshDay As Integer = -1
65     Protected m_excdThreshNight As Integer = -1
66     Protected m_timerRun1 As String = ""
67     Protected m_timerStop1 As String = ""
68     Protected m_timerChanged As Boolean
69
70
71     Public Enum Owner
72         Aberdeen
73         CERL
74         Nobody
75     End Enum
76
77
78 #Region "Properties"
79     Public Property InstallPath() As String
80         Get
81             Return m_installPath
82         End Get
83         Set(ByVal value As String)
84             m_installPath = value
85         End Set
86     End Property
87
88     Public Property IncludeR() As Boolean
89         Get
90             Return m_includeR
91         End Get
92         Set(ByVal value As Boolean)
```

```vb
 93                     m_includeR = value
 94                 End Set
 95             End Property
 96             Public Property IncludeLC() As Boolean
 97                 Get
 98                     Return m_includeLC
 99                 End Get
100                 Set(ByVal value As Boolean)
101                     m_includeLC = value
102                 End Set
103             End Property
104             Public Property IncludeQ() As Boolean
105                 Get
106                     Return m_includeQ
107                 End Get
108                 Set(ByVal value As Boolean)
109                     m_includeQ = value
110                 End Set
111             End Property
112             Public Property AllowCallIns() As Boolean
113                 Get
114                     Return m_allowCallIns
115                 End Get
116                 Set(ByVal value As Boolean)
117                     m_allowCallIns = value
118                 End Set
119             End Property
120             Public Property ResetDataYN() As Boolean
121                 Get
122                     Return m_resetDataYN
123                 End Get
124                 Set(ByVal value As Boolean)
125                     m_resetDataYN = value
126                 End Set
127             End Property
128             Public Property ResetTimeYN() As Boolean
129                 Get
130                     Return m_resetTimeYN
131                 End Get
132                 Set(ByVal value As Boolean)
133                     m_resetTimeYN = value
134                 End Set
135             End Property
136             Public Property IsDownloadDone() As Boolean
137                 Get
138                     Return m_isDownloadDone
139                 End Get
140                 Set(ByVal value As Boolean)
141                     m_isDownloadDone = value
142                 End Set
143             End Property
144             Public Property IsDownloadInProgress() As Boolean
145                 Get
146                     Return m_isDownloadInProgress
147                 End Get
148                 Set(ByVal value As Boolean)
149                     m_isDownloadInProgress = value
150                 End Set
151             End Property
152             Public Property IsEnabled() As Boolean
153                 Get
154                     Return m_isEnabled
155                 End Get
156                 Set(ByVal value As Boolean)
157                     m_isEnabled = value
158                 End Set
159             End Property
160             Public Property DLTries() As Integer
161                 Get
162                     Return m_dlTries
163                 End Get
164                 Set(ByVal value As Integer)
165                     m_dlTries = value
166                 End Set
167             End Property
168             Public Property DialTries() As Integer
169                 Get
170                     Return m_dialTries
171                 End Get
172                 Set(ByVal value As Integer)
173                     m_dialTries = value
174                 End Set
175             End Property
176             Public Property Initialize() As Boolean
177                 Get
178                     Return m_initialize
179                 End Get
180                 Set(ByVal value As Boolean)
181                     m_initialize = value
182                 End Set
183             End Property
184             Public Property SendCustP() As Boolean
```

```vb
185            Get
186                Return m_sendCustP
187            End Get
188            Set(ByVal value As Boolean)
189                m_sendCustP = value
190            End Set
191        End Property
192
193
194        Public Property UnitSerial() As String
195            Get
196                Return m_unitSerial
197            End Get
198            Set(ByVal value As String)
199                m_unitSerial = value
200            End Set
201        End Property
202        Public Property UnitNum() As String
203            Get
204                Return m_unitLNum
205            End Get
206            Set(ByVal value As String)
207                m_unitLNum = value
208            End Set
209        End Property
210        Public Property LockCode() As String
211            Get
212                Return m_lockCode
213            End Get
214            Set(ByVal value As String)
215                m_lockCode = value
216            End Set
217        End Property
218        Public Property UnitLocation() As String
219            Get
220                Return m_unitLocation
221            End Get
222            Set(ByVal value As String)
223                m_unitLocation = value
224            End Set
225        End Property
226        Public Property UnitPhoneNum() As String
227            Get
228                Return m_unitPhoneNum
229            End Get
230            Set(ByVal value As String)
231                m_unitPhoneNum = value
232            End Set
233        End Property
234        Public Property UnitOwner() As Owner
235            Get
236                Return m_unitOwner
237            End Get
238            Set(ByVal value As Owner)
239                m_unitOwner = value
240            End Set
241        End Property
242        Public Property LastDL() As DateTime
243            Get
244                Return m_lastDL
245            End Get
246            Set(ByVal value As DateTime)
247                m_lastDL = value
248            End Set
249        End Property
250        Public Property CustParams() As String
251            Get
252                Return m_custParams
253            End Get
254            Set(ByVal value As String)
255                m_custParams = value
256            End Set
257        End Property
258
259
260        Public Property NumExceedances() As Integer
261            Get
262                Return m_numExceedances
263            End Get
264            Set(ByVal value As Integer)
265                m_numExceedances = value
266            End Set
267        End Property
268        Public Property NumIntervals() As Integer
269            Get
270                Return m_numIntervals
271            End Get
272            Set(ByVal value As Integer)
273                m_numIntervals = value
274            End Set
275        End Property
276        Public Property NumStartStops() As Integer
```

```vb
277                Get
278                    Return m_numStartStops
279                End Get
280                Set(ByVal value As Integer)
281                    m_numStartStops = value
282                End Set
283            End Property
284            Public Property NumCalibrations() As Integer
285                Get
286                    Return m_numCalibrations
287                End Get
288                Set(ByVal value As Integer)
289                    m_numCalibrations = value
290                End Set
291            End Property
292            Public Property BattVoltage() As Double
293                Get
294                    Return m_battVoltage
295                End Get
296                Set(ByVal value As Double)
297                    m_battVoltage = value
298                End Set
299            End Property
300            Public Property ErrorString() As String
301                Get
302                    Return m_errorString
303                End Get
304                Set(ByVal value As String)
305                    m_errorString = value
306                End Set
307            End Property
308            Public Property EList() As List(Of String)
309                Get
310                    Return m_eList
311                End Get
312                Set(ByVal value As List(Of String))
313                    m_eList = value
314                End Set
315            End Property
316            Public Property IList() As List(Of String)
317                Get
318                    Return m_iList
319                End Get
320                Set(ByVal value As List(Of String))
321                    m_iList = value
322                End Set
323            End Property
324            Public Property RList() As List(Of String)
325                Get
326                    Return m_rList
327                End Get
328                Set(ByVal value As List(Of String))
329                    m_rList = value
330                End Set
331            End Property
332            Public Property QList() As List(Of String)
333                Get
334                    Return m_qList
335                End Get
336                Set(ByVal value As List(Of String))
337                    m_qList = value
338                End Set
339            End Property
340            Public Property LList() As List(Of String)
341                Get
342                    Return m_lList
343                End Get
344                Set(ByVal value As List(Of String))
345                    m_lList = value
346                End Set
347            End Property
348            Public Property CList() As List(Of String)
349                Get
350                    Return m_cList
351                End Get
352                Set(ByVal value As List(Of String))
353                    m_cList = value
354                End Set
355            End Property
356            Public Property ExcdThreshold() As Integer
357                Get
358                    Return m_excdThreshold
359                End Get
360                Set(ByVal value As Integer)
361                    m_excdThreshold = value
362                End Set
363            End Property
364            Public Property CalLevel() As Double
365                Get
366                    Return m_calLevel
367                End Get
368                Set(ByVal value As Double)
```

```vb
369                     m_calLevel = value
370                 End Set
371         End Property
372         Public Property ExcdDay() As Integer
373             Get
374                     Return m_excdThreshDay
375             End Get
376             Set(ByVal value As Integer)
377                     m_excdThreshDay = value
378             End Set
379         End Property
380         Public Property ExcdNight() As Integer
381             Get
382                     Return m_excdThreshNight
383             End Get
384             Set(ByVal value As Integer)
385                     m_excdThreshNight = value
386             End Set
387         End Property
388         Public Property TimerRun1() As String
389             Get
390                     Return m_timerRun1
391             End Get
392             Set(ByVal value As String)
393                     m_timerRun1 = value
394             End Set
395         End Property
396         Public Property TimerStop1() As String
397             Get
398                     Return m_timerStop1
399             End Get
400             Set(ByVal value As String)
401                     m_timerStop1 = value
402             End Set
403         End Property
404         Public Property TimerChanged() As Boolean
405             Get
406                     Return m_timerChanged
407             End Get
408             Set(ByVal value As Boolean)
409                     m_timerChanged = value
410             End Set
411         End Property
412
413
414  #End Region
415
416  #Region "Constructors"
417         Public Sub New()
418             Me.InstallPath = "C:\Program Files\L-D Download\"
419             Me.IncludeR = False
420             Me.IncludeLC = False
421             Me.IncludeQ = False
422             Me.AllowCallIns = False
423             Me.ResetDataYN = False
424             Me.ResetTimeYN = False
425             Me.IsDownloadDone = False
426             Me.IsDownloadInProgress = False
427             Me.IsEnabled = True
428             Me.DLTries = 0
429             Me.DialTries = 0
430             Me.Initialize = False
431             Me.SendCustP = False
432             Me.UnitLocation = ""
433             Me.UnitNum = "null"
434             Me.LastDL = New Date(1900, 1, 1, 0, 0, 0)
435             Me.ExcdDay = Form1.appset.WorkdayExcdThresh
436             Me.ExcdNight = Form1.appset.OtherExcdThresh
437             Me.TimerRun1 = "06:00"
438             Me.TimerStop1 = "22:00"
439             Me.TimerChanged = False
440         End Sub
441         Public Sub New(ByVal unitnum As String)
442             Me.InstallPath = "C:\Program Files\L-D Download\"
443             Me.IncludeR = False
444             Me.IncludeLC = False
445             Me.IncludeQ = False
446             Me.AllowCallIns = False
447             Me.ResetDataYN = False
448             Me.ResetTimeYN = False
449             Me.IsDownloadDone = False
450             Me.IsDownloadInProgress = False
451             Me.IsEnabled = True
452             Me.DLTries = 0
453             Me.DialTries = 0
454             Me.Initialize = False
455             Me.SendCustP = False
456             Me.UnitNum = unitnum
457             Me.UnitLocation = ""
458             Me.UnitOwner = Owner.Nobody
459             Me.LastDL = New Date(1900, 1, 1, 0, 0, 0)
460             Select Case Me.UnitOwner
```

```vb
461                Case Owner.Aberdeen : Me.LockCode = "22222222"
462                Case Owner.CERL : Me.LockCode = "22222222"
463                Case Owner.Nobody : Me.LockCode = "22222222"
464            End Select
465            Me.ExcdDay = Form1.appset.WorkdayExcdThresh
466            Me.ExcdNight = Form1.appset.OtherExcdThresh
467            Me.TimerRun1 = "06:00"
468            Me.TimerStop1 = "22:00"
469            Me.TimerChanged = False
470        End Sub
471        Public Sub New(ByVal unitnum As String, ByVal unitphonenum As String, ByVal uowner As Owner)
472            Me.InstallPath = "C:\Program Files\L-D Download\"
473            Me.IncludeR = False
474            Me.IncludeLC = False
475            Me.IncludeQ = False
476            Me.AllowCallIns = False
477            Me.ResetDataYN = False
478            Me.ResetTimeYN = False
479            Me.IsDownloadDone = False
480            Me.IsDownloadInProgress = False
481            Me.IsEnabled = True
482            Me.DLTries = 0
483            Me.DialTries = 0
484            Me.Initialize = False
485            Me.SendCustP = False
486            Me.UnitNum = unitnum
487            Me.UnitLocation = ""
488            Me.UnitPhoneNum = unitphonenum
489            Me.UnitOwner = uowner
490            Me.LastDL = New Date(1900, 1, 1, 0, 0, 0)
491            Select Case Me.UnitOwner
492                Case Owner.Aberdeen : Me.LockCode = "22222222"
493                Case Owner.CERL : Me.LockCode = "22222222"
494                Case Owner.Nobody : Me.LockCode = "22222222"
495            End Select
496            Me.ExcdDay = Form1.appset.WorkdayExcdThresh
497            Me.ExcdNight = Form1.appset.OtherExcdThresh
498            Me.TimerRun1 = "06:00"
499            Me.TimerStop1 = "22:00"
500            Me.TimerChanged = False
501        End Sub
502        Public Sub New(ByVal unitnum As String, ByVal unitphonenum As String, ByVal uowner As Owner, ByVal uLoc ↙
           As String)
503            Me.InstallPath = "C:\Program Files\L-D Download\"
504            Me.IncludeR = False
505            Me.IncludeLC = False
506            Me.IncludeQ = False
507            Me.AllowCallIns = False
508            Me.ResetDataYN = False
509            Me.ResetTimeYN = False
510            Me.IsDownloadDone = False
511            Me.IsDownloadInProgress = False
512            Me.IsEnabled = True
513            Me.DLTries = 0
514            Me.DialTries = 0
515            Me.Initialize = False
516            Me.SendCustP = False
517            Me.UnitNum = unitnum
518            Me.UnitLocation = ""
519            Me.UnitPhoneNum = unitphonenum
520            Me.UnitOwner = uowner
521            Me.UnitLocation = uLoc
522            Me.LastDL = New Date(1900, 1, 1, 0, 0, 0)
523            Select Case Me.UnitOwner
524                Case Owner.Aberdeen : Me.LockCode = "22222222"
525                Case Owner.CERL : Me.LockCode = "22222222"
526                Case Owner.Nobody : Me.LockCode = "22222222"
527            End Select
528            Me.ExcdDay = Form1.appset.WorkdayExcdThresh
529            Me.ExcdNight = Form1.appset.OtherExcdThresh
530            Me.TimerRun1 = "06:00"
531            Me.TimerStop1 = "22:00"
532            Me.TimerChanged = False
533        End Sub
534 #End Region
535
536 #Region "Methods"
537
538    Public Function SerializeReferenceTime(ByVal u1 As Date, ByVal h1 As Date, ByVal o1 As Double) As String
539        'exports the time data taken when the unit was downloaded
540        'stores the data as a serialized object
541
542        Dim RT As ReferenceTime = New ReferenceTime(u1, h1, o1)
543
544        Using fs As New FileStream(Me.InstallPath & "Reference Times\" & _
545        Me.UnitNum & ".dat", FileMode.Create)
546
547            Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
548            bf.Serialize(fs, RT)
549        End Using
550
551        'return the unique part of the filename
```

```vb
552             Return Me.UnitNum & ".dat" '& "_" & h1.ToLongDateString
553
554     End Function
555
556     Public Function SerializeReferenceTime(ByVal RT As ReferenceTime) As String
557         'exports the time data taken when the unit was downloaded
558         'stores the data as a serialized object
559
560         Using fs As New FileStream(Me.InstallPath & "Reference Times\" & _
561         Me.UnitNum & ".dat", FileMode.Create)
562
563             Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
564             bf.Serialize(fs, RT)
565         End Using
566
567         'return the unique part of the filename
568         Return Me.UnitNum & ".dat" '& "_" & RT.HostTimeZero.ToLongDateString
569
570     End Function
571
572     Public Function GetReferenceTime() As ReferenceTime
573         'imports a serialized object which contains the previously recorded time
574         Dim RT As ReferenceTime
575
576         Try
577             Using fs As New FileStream(Me.InstallPath & "Reference Times\" & Me.UnitNum _
578                     & ".dat", FileMode.Open)
579
580                 Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
581                 RT = DirectCast(bf.Deserialize(fs), ReferenceTime)
582
583             End Using
584         Catch ex As FileNotFoundException   'file does not exist
585             RT = New ReferenceTime(New Date(1), New Date(1), 0)
586
587         End Try
588
589         Return RT
590
591     End Function
592
593     Public Sub SerializeReports()
594         'Using sw As New StreamWriter(Me.InstallPath & "Reports\" & Me.UnitNum & " " & _
595         'Date.Now.ToString("dd-MMM-yy HH_mm_ss") & ".txt", True)
596
597         '    For Each item As String In EList
598         '        sw.WriteLine(item)
599         '    Next
600         '    For Each item As String In IList
601         '        sw.WriteLine(item)
602         '    Next
603
604
605         '    sw.WriteLine()
606         '    sw.WriteLine(BattVoltage & " V")
607         '    sw.WriteLine(ErrorString)
608         '    sw.WriteLine()
609         '    sw.WriteLine("----------------------------------------------------------------------")
610
611         'End Using
612         If Me.IncludeR OrElse Me.IncludeQ OrElse Me.IncludeLC Then
613             Using sw2 As New StreamWriter(Me.InstallPath & "Setup Logs\" & Me.UnitNum & " Setup " & _
614             Date.Now.ToString("dd-MMM-yy HH_mm_ss") & ".txt", True)
615                 For Each item As String In RList
616                     sw2.WriteLine(item)
617                 Next
618                 'For Each item As String In QList
619                 '    sw2.WriteLine(item)
620                 'Next
621                 For Each item As String In LList
622                     sw2.WriteLine(item)
623                 Next
624                 For Each item As String In CList
625                     sw2.WriteLine(item)
626                 Next
627
628             End Using
629         End If
630     End Sub
631
632     Public Function SaveToDB(ByVal RT As ReferenceTime) As String
633         'saves report info to a database
634         'returns an error string if an SQL error occurred
635
636         Dim conn As New SqlConnection("server=localhost; database=Complaint; Integrated Security=SSPI")
637         Dim sep() As Char = {","c}
638         Dim sep2() As Char = {" "c}
639         Dim trim1() As Char = {"'"c}
640         Dim eCols As String = "UnitID,Date_Time,Duration,CLeq,CSEL,CLmax,CPeak,UwPeak,Counts,[Load],Comments↙
        ,Symmetry,Wind_Speed,Max_Wind_Speed,Wind_Direction,Temperature,Max_Temperature,Relative_Humidity,↙
        Max_Relative_Humidity"
641         Dim iCols As String = "UnitID,Date_Time,Interval_End_Time,CLeq,CSEL,CLmin,CLmax,CPeak,UwPeak,↙
```

```vb
          Counts_a,Counts_b,Counts_c,Unknown,[L(5)],[L(5)_value],[L(10)],[L(10)_value],[L(33)],[L(33)_value],[L  ↙
          (50)],[L(50)_value],[L(67)],[L(67)_value],[L(90)],[L(90)_value],Average_Wind_Speed,Max_Wind_Speed,      ↙
          Wind_Direction,Average_Temperature,Min_Temperature,Max_Temperature,Average_Relative_Humidity,          ↙
          Min_Relative_Humidity,Max_Relative_Humidity"
642
643        Dim ctCol As String = ",Corrected_Time"
644        Dim hrCol As String = ",TotalHrs"
645        Dim hrCCol As String = ",TotalHrs_Corrected"
646        Dim cmd As New SqlCommand("", conn)
647        Dim errorStr As String = ""
648        'Dim qCols As New System.Text.StringBuilder
649        Dim qRow As New System.Text.StringBuilder
650        Dim tryp1 As Boolean = False
651        Dim tryp2 As Boolean = False
652
653        Try
654            conn.Open()
655            For Each row As String In Me.EList
656                Dim cells() As String = row.Split(sep, StringSplitOptions.None)
657                row = ""
658                'cells will have 20 elements if corrected time is included, 19 if not
659                '(they have been error-checked by now)
660
661                'replace "E   1" with the unit number
662                cells(0) = Me.UnitNum
663
664                For i As Integer = 0 To cells.Length - 1
665                    cells(i) = cells(i).Trim()
666                    If i = 0 OrElse i = 1 OrElse i = 2 OrElse i = 10 OrElse i = 14 OrElse i = 19 Then
667                        cells(i) = "'" & cells(i) & "'"
668                    End If
669                    row &= cells(i)
670                    If i < cells.Length - 1 Then
671                        row &= ", "
672                    End If
673                Next
674
675                'append Total Hours and Corrected Total Hrs to the row string
676                'Dim dt As Date = Date.Now
677                'cells(1) = cells(1).Trim(trim1)
678                'tryp1 = Date.TryParse(cells(1), dt)
679                'If tryp1 Then
680                '    Dim dt2 As New TimeSpan(dt.Ticks)
681                '    Dim hrs As Double = dt2.TotalHours
682                '    hrs = Math.Round(hrs, 2)
683                '    row &= ", " & hrs.ToString
684                'End If
685
686                'If cells.Length = 20 Then 'append the total # of hours for the corrected time as well
687                '    cells(19) = cells(19).Trim(trim1)
688                '    tryp2 = Date.TryParse(cells(19), dt)
689                '    If tryp2 Then
690                '        Dim dt2 As New TimeSpan(dt.Ticks)
691                '        Dim hrs As Double = dt2.TotalHours
692                '        hrs = Math.Round(hrs, 2)
693                '        row &= ", " & hrs.ToString
694                '    End If
695                'End If
696
697                'MessageBox.Show(row)
698
699                'make the query string
700                If cells.Length = 20 Then
701                    cmd.CommandText = "INSERT INTO EventData (" & eCols & ctCol & ") VALUES (" & row & ")"
702                Else
703                    cmd.CommandText = "INSERT INTO EventData (" & eCols & ") VALUES (" & row & ")"
704                End If
705
706                'If cells.Length = 20 Then
707                '    If tryp1 AndAlso tryp2 Then
708                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & ctCol & hrCol & hrCCol & _
709                '                          ") VALUES (" & row & ")"
710                '    ElseIf tryp2 Then
711                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & ctCol & hrCCol & _
712                '                          ") VALUES (" & row & ")"
713                '    ElseIf tryp1 Then
714                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & ctCol & hrCol & _
715                '                          ") VALUES (" & row & ")"
716                '    Else
717                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & ctCol & _
718                '                          ") VALUES (" & row & ")"
719                '    End If
720                'Else
721                '    If tryp1 Then
722                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & hrCol & ") VALUES (" & row & ↙
       ")"
723                '    Else
724                '        cmd.CommandText = "INSERT INTO EventData (" & eCols & ") VALUES (" & row & ")"
725                '    End If
726                'End If
727
728                cmd.ExecuteNonQuery()
```

```vb
729               Next
730
731               For Each row As String In IList
732                   Dim cells() As String = row.Split(sep, StringSplitOptions.None)
733                   row = ""
734                   'cells will have 35 elements if corrected time is included, 34 if not
735                   '(they have been error-checked by now)
736
737                   'replace "I   1" with the unit number
738                   cells(0) = Me.UnitNum
739
740                   For i As Integer = 0 To cells.Length - 1
741                       cells(i) = cells(i).Trim()
742                       If i = 0 OrElse i = 1 OrElse i = 2 OrElse i = 27 OrElse i = 34 Then
743                           cells(i) = "'" & cells(i) & "'"
744                       End If
745                       row &= cells(i)
746                       If i < cells.Length - 1 Then
747                           row &= ", "
748                       End If
749                   Next
750
751                   'append Total Hours and Corrected Total Hrs to the row string
752                   'Dim dt As Date = Date.Now
753                   'cells(1) = cells(1).Trim(trim1)
754                   'tryp1 = Date.TryParse(cells(1), dt)
755                   'If tryp1 Then
756                   '    Dim dt2 As New TimeSpan(dt.Ticks)
757                   '    Dim hrs As Double = dt2.TotalHours
758                   '    hrs = Math.Round(hrs, 2)
759                   '    row &= ", " & hrs.ToString
760                   'End If
761
762                   'If cells.Length = 35 Then 'append the total # of hours for the corrected time as well
763                   '    cells(34) = cells(34).Trim(trim1)
764                   '    tryp2 = Date.TryParse(cells(34), dt)
765                   '    If tryp2 Then
766                   '        Dim dt2 As New TimeSpan(dt.Ticks)
767                   '        Dim hrs As Double = dt2.TotalHours
768                   '        hrs = Math.Round(hrs, 2)
769                   '        row &= ", " & hrs.ToString
770                   '    End If
771                   'End If
772
773                   'make the query string
774                   If cells.Length = 35 Then
775                       cmd.CommandText = "INSERT INTO Intervals (" & iCols & ctCol & ") VALUES (" & row & ")"
776                   Else
777                       cmd.CommandText = "INSERT INTO Intervals (" & iCols & ") VALUES (" & row & ")"
778                   End If
779
780                   'If cells.Length = 35 Then
781                   '    If tryp1 AndAlso tryp2 Then
782                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & ctCol & hrCol & hrCCol & _
783                   '                          ") VALUES (" & row & ")"
784                   '    ElseIf tryp2 Then
785                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & ctCol & hrCCol & _
786                   '                          ") VALUES (" & row & ")"
787                   '    ElseIf tryp1 Then
788                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & ctCol & hrCol & _
789                   '                          ") VALUES (" & row & ")"
790                   '    Else
791                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & ctCol & ") VALUES (" & row & ↵
        ")"
792                   '    End If
793                   'Else
794                   '    If tryp1 Then
795                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & hrCol & ") VALUES (" & row & ↵
        ")"
796                   '    Else
797                   '        cmd.CommandText = "INSERT INTO Intervals (" & iCols & ") VALUES (" & row & ")"
798                   '    End If
799                   'End If
800
801                   cmd.ExecuteNonQuery()
802               Next
803
804               'save Q's to Q table - entire setup file becomes one row
805               If Me.IncludeQ Then
806                   qRow.Append("'" & Me.UnitNum & "', '" & RT.HostTimeZero & "', ")
807                   For Each line As String In QList
808                       Dim cells() As String = line.Split(sep, StringSplitOptions.None)
809                       Dim datacell As String = ""
810                       'Dim qcell As String = ""
811                       'cells(0) = cells(0).Trim()
812                       ''If there are additional commas on this line, combine the different cells into one data↵
        line
813                       For j As Integer = 1 To cells.Length - 1
814                           datacell &= cells(j)
815                       Next
816                       cells(1) = datacell
817                       cells(1) = cells(1).Trim()
```

```vb
818
819                        'remove spaces between "Q  2"
820
821                        'Dim Q() As String = cells(0).Split(sep2, StringSplitOptions.RemoveEmptyEntries)
822                        'For j As Integer = 0 To Q.Length - 1
823                        '    qcell &= Q(j)
824                        'Next
825                        'qCols.Append(cells(0) & ",")
826                        qRow.Append("'" & cells(1) & "', ")
827                    Next
828                    'qCols.Remove(qCols.Length - 1, 1)
829                    qRow.Remove(qRow.Length - 2, 2)
830
831                    'append Total Hours to the corrected string
832                    'Dim dt2 As New TimeSpan(RT.HostTimeZero.Ticks)
833                    'Dim hrs As Double = dt2.TotalHours
834                    'hrs = Math.Round(hrs, 2)
835                    'qRow.Append("," & hrs.ToString)
836
837                    'make the query string
838                    cmd.CommandText = "INSERT INTO Q_Params VALUES (" & qRow.ToString() & ")"
839                    'OK to not put column names - the RecordNum identifier gets skipped over automatically
840
841                    cmd.ExecuteNonQuery()
842                End If
843
844                'record the new # of E rows to the table DB_Size
845                'this is needed to make the database viewer update more efficiently
846
847                cmd.CommandText = "SELECT count([Load]) FROM EventData"
848                Dim numEDRows As Long = CLng(cmd.ExecuteScalar())
849                cmd.CommandText = "UPDATE DB_Size SET NumEDRows = " & numEDRows.ToString & " WHERE ID = 1"
850                cmd.ExecuteNonQuery()
851
852          Catch ex As SqlException
853                errorStr = "SQL Error: " & ex.ErrorCode & " - " & ex.Message
854          Finally
855                conn.Close()
856          End Try
857
858          Return errorStr
859
860      End Function
861 #End Region
862
863 End Class
864
865
866
867 <Serializable()> _
868 Public Class ReferenceTime
869      Protected m_unitTimeAvg1 As Date
870      Protected m_hostTimeAvg1 As Date
871      Protected m_tOffsetAvg1 As Double
872
873
874      Public Property UnitTimeZero() As Date
875          Get
876                Return m_unitTimeAvg1
877          End Get
878          Set(ByVal value As Date)
879                m_unitTimeAvg1 = value
880          End Set
881      End Property
882      Public Property HostTimeZero() As Date
883          Get
884                Return m_hostTimeAvg1
885          End Get
886          Set(ByVal value As Date)
887                m_hostTimeAvg1 = value
888          End Set
889      End Property
890      Public Property TOffsetAvg1() As Double
891          Get
892                Return m_tOffsetAvg1
893          End Get
894          Set(ByVal value As Double)
895                m_tOffsetAvg1 = value
896          End Set
897      End Property
898
899
900      Public Sub New(ByVal u1 As Date, ByVal h1 As Date, ByVal o1 As Double)
901
902          Me.UnitTimeZero = u1
903          Me.HostTimeZero = h1
904          Me.TOffsetAvg1 = o1
905      End Sub
906
907 End Class
908
909 <Serializable()> _
```

```vb
910 Public Class AppSettings
911
912     Protected m_installPath As String
913     Protected m_dialAttempts As Integer
914     Protected m_modem1num As String
915     Protected m_modem2num As String
916     Protected m_modem3num As String
917     Protected m_modem4num As String
918     Protected m_isSatAWorkday As Boolean
919     Protected m_isSunAWorkday As Boolean
920     Protected m_custParams As String
921     Protected m_workdayExcdThresh As Integer
922     Protected m_otherExcdThresh As Integer
923     Protected m_dialInProg As Boolean
924     Protected m_saveLogAfterDL As Boolean
925     Protected m_modemPorts(3) As String
926     Protected m_stopAtNight As Boolean
927     Protected m_conn9600 As Boolean
928     Protected m_maxLines As Integer
929
930
931     Protected m_weeknightDL As TimeSpan
932     Protected m_weekendDL As TimeSpan
933     Protected m_workdayStart As TimeSpan
934     Protected m_workdayEnd As TimeSpan
935
936 #Region "Properties"
937     Public Property InstallPath() As String
938         Get
939             Return m_installPath
940         End Get
941         Set(ByVal value As String)
942             m_installPath = value
943         End Set
944     End Property
945     Public Property DialAttempts() As Integer
946         Get
947             Return m_dialAttempts
948         End Get
949         Set(ByVal value As Integer)
950             m_dialAttempts = value
951         End Set
952     End Property
953     Public Property Modem1Num() As String
954         Get
955             Return m_modem1num
956         End Get
957         Set(ByVal value As String)
958             m_modem1num = value
959         End Set
960     End Property
961     Public Property Modem2Num() As String
962         Get
963             Return m_modem2num
964         End Get
965         Set(ByVal value As String)
966             m_modem2num = value
967         End Set
968     End Property
969     Public Property Modem3Num() As String
970         Get
971             Return m_modem3num
972         End Get
973         Set(ByVal value As String)
974             m_modem3num = value
975         End Set
976     End Property
977     Public Property Modem4Num() As String
978         Get
979             Return m_modem4num
980         End Get
981         Set(ByVal value As String)
982             m_modem4num = value
983         End Set
984     End Property
985     Public Property IsSatAWorkday() As Boolean
986         Get
987             Return Me.m_isSatAWorkday
988         End Get
989         Set(ByVal value As Boolean)
990             Me.m_isSatAWorkday = value
991         End Set
992     End Property
993     Public Property IsSunAWorkday() As Boolean
994         Get
995             Return Me.m_isSunAWorkday
996         End Get
997         Set(ByVal value As Boolean)
998             Me.m_isSunAWorkday = value
999         End Set
1000    End Property
1001    Public Property CustParams() As String
```

```vb
1002            Get
1003                Return m_custParams
1004            End Get
1005            Set(ByVal value As String)
1006                m_custParams = value
1007            End Set
1008        End Property
1009        Public Property WorkdayExcdThresh() As Integer
1010            Get
1011                Return m_workdayExcdThresh
1012            End Get
1013            Set(ByVal value As Integer)
1014                Me.m_workdayExcdThresh = value
1015            End Set
1016        End Property
1017        Public Property OtherExcdThresh() As Integer
1018            Get
1019                Return Me.m_otherExcdThresh
1020            End Get
1021            Set(ByVal value As Integer)
1022                Me.m_otherExcdThresh = value
1023            End Set
1024        End Property
1025        Public Property DialInProg() As Boolean
1026            Get
1027                Return Me.m_dialInProg
1028            End Get
1029            Set(ByVal value As Boolean)
1030                Me.m_dialInProg = value
1031            End Set
1032        End Property
1033        Public Property SaveLogAfterDL() As Boolean
1034            Get
1035                Return Me.m_saveLogAfterDL
1036            End Get
1037            Set(ByVal value As Boolean)
1038                Me.m_saveLogAfterDL = value
1039            End Set
1040        End Property
1041        Public Property ModemPorts() As String()
1042            Get
1043                Return m_modemPorts
1044            End Get
1045            Set(ByVal value() As String)
1046                m_modemPorts = value
1047            End Set
1048        End Property
1049        Public Property Modem1Port() As String
1050            Get
1051                Return m_modemPorts(0)
1052            End Get
1053            Set(ByVal value As String)
1054                m_modemPorts(0) = value
1055            End Set
1056        End Property
1057        Public Property Modem2Port() As String
1058            Get
1059                Return m_modemPorts(1)
1060            End Get
1061            Set(ByVal value As String)
1062                m_modemPorts(1) = value
1063            End Set
1064        End Property
1065        Public Property Modem3Port() As String
1066            Get
1067                Return m_modemPorts(2)
1068            End Get
1069            Set(ByVal value As String)
1070                m_modemPorts(2) = value
1071            End Set
1072        End Property
1073        Public Property Modem4Port() As String
1074            Get
1075                Return m_modemPorts(3)
1076            End Get
1077            Set(ByVal value As String)
1078                m_modemPorts(3) = value
1079            End Set
1080        End Property
1081        Public Property StopAtNight() As Boolean
1082            Get
1083                Return m_stopAtNight
1084            End Get
1085            Set(ByVal value As Boolean)
1086                m_stopAtNight = value
1087            End Set
1088        End Property
1089        Public Property Conn9600() As Boolean
1090            Get
1091                Return m_conn9600
1092            End Get
1093            Set(ByVal value As Boolean)
```

```vb
1094                    m_conn9600 = value
1095                End Set
1096        End Property
1097        Public Property MaxLines() As Integer
1098            Get
1099                Return m_maxLines
1100            End Get
1101            Set(ByVal value As Integer)
1102                If value > 19 Then
1103                    m_maxLines = value
1104                Else
1105                    m_maxLines = 1000
1106                End If
1107            End Set
1108        End Property
1109
1110
1111        Public Property WeeknightDL() As TimeSpan
1112            Get
1113                Return m_weeknightDL
1114            End Get
1115            Set(ByVal value As TimeSpan)
1116                m_weeknightDL = value
1117            End Set
1118        End Property
1119        Public Property WeekendDL() As TimeSpan
1120            Get
1121                Return m_weekendDL
1122            End Get
1123            Set(ByVal value As TimeSpan)
1124                m_weekendDL = value
1125            End Set
1126        End Property
1127        Public Property WorkdayStart() As TimeSpan
1128            Get
1129                Return m_workdayStart
1130            End Get
1131            Set(ByVal value As TimeSpan)
1132                m_workdayStart = value
1133            End Set
1134        End Property
1135        Public Property WorkdayEnd() As TimeSpan
1136            Get
1137                Return m_workdayEnd
1138            End Get
1139            Set(ByVal value As TimeSpan)
1140                m_workdayEnd = value
1141            End Set
1142        End Property
1143 #End Region
1144
1145
1146        Public Sub New()
1147            With Me
1148                .InstallPath = "C:\Program Files\L-D Download\"
1149                .DialAttempts = 4
1150                .Modem1Num = "14102721390"
1151                .Modem2Num = "14102721390"
1152                .Modem3Num = "14102721390"
1153                .Modem4Num = "14102721390"
1154                .Modem1Port = "COM10"
1155                .Modem2Port = "COM11"
1156                .Modem3Port = "COM12"
1157                .Modem4Port = "COM13"
1158                .IsSatAWorkday = True
1159                .IsSunAWorkday = False
1160                .StopAtNight = True
1161                .CustParams = ""
1162                .WorkdayExcdThresh = 110
1163                .OtherExcdThresh = 110
1164                .DialInProg = False
1165                .SaveLogAfterDL = True
1166                .Conn9600 = True
1167                .MaxLines = 1000
1168                .WeeknightDL = New TimeSpan(4, 30, 0)
1169                .WeekendDL = New TimeSpan(4, 30, 0)
1170                .WorkdayStart = New TimeSpan(6, 0, 0)
1171                .WorkdayEnd = New TimeSpan(22, 0, 0)
1172            End With
1173        End Sub
1174
1175        Public Sub Serialize()
1176
1177            Using fs As New FileStream(Me.InstallPath & "app.dat", FileMode.Create)
1178
1179                Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
1180                bf.Serialize(fs, Me)
1181            End Using
1182
1183        End Sub
1184
1185 End Class
```

```vb
1  'Modem
2  'Instantiate one for each modem attached to a system
3
4  'Manages the transfer of data to and from each unit
5
6
7  Imports System
8  Imports System.IO.Ports
9  Imports System.Text
10 Imports System.Text.RegularExpressions
11 Imports System.Collections.Generic
12 Imports System.Threading
13 Imports System.ComponentModel
14
15
16 Public Class Modem
17
18     Protected WithEvents comx As New SerialPort
19     Protected m_modemMode As Mode
20     Public modemPhoneNum As String = ""
21     Public speakerOn As Boolean
22     Private getUnitTimeCounter As Integer = 0
23
24     Public Event UpdateOutBoxesEvent As EventHandler(Of UpdateOutBoxesEventArgs)
25     'Public Event UpdateStatusBar As EventHandler(Of StatusBarEventArgs)
26     Public Event SBUnitCounter As EventHandler(Of StatusBarEventArgs)
27     Public Event UpdateMessages As EventHandler(Of StatusBarEventArgs)
28
29
30     Public Property modemMode() As Mode
31         Get
32             Return m_modemMode
33         End Get
34         Set(ByVal value As Mode)
35             m_modemMode = value
36         End Set
37     End Property
38
39     Public Sub New(ByVal portName As String, ByVal phoneNum As String)
40         With comx
41             .PortName = portName
42             .BaudRate = 115200
43             .Parity = IO.Ports.Parity.None
44             .DataBits = 8
45             .StopBits = IO.Ports.StopBits.One
46             .Encoding = System.Text.Encoding.UTF8
47             .DtrEnable = True    'needed to make MultiModem work
48             .RtsEnable = True    'needed to make MultiModem receive text
49         End With
50         modemPhoneNum = phoneNum
51     End Sub
52
53     Public Sub New(ByVal portName As String, ByVal phoneNum As String, ByVal classMode As Mode)
54         With comx
55             .PortName = portName
56             .BaudRate = 115200
57             .Parity = IO.Ports.Parity.None
58             .DataBits = 8
59             .StopBits = IO.Ports.StopBits.One
60             .Encoding = System.Text.Encoding.UTF8
61             .DtrEnable = True    'needed to make MultiModem work
62             .RtsEnable = True    'needed to make MultiModem receive text
63         End With
64         modemPhoneNum = phoneNum
65         m_modemMode = classMode
66     End Sub
67
68     Public Enum Mode
69         Listen
70         Dial
71         Standby
72         ListenCancel
73     End Enum
74
75 #Region "Read/Write Engine"
76
77     Public Function ReadLine(ByVal searchChar As Char) As String
78         'reads until the search char is found
79         'faster than ReadUntilChar
80         'also returns nothing if the search character is not found
81
82         Dim returnstr As String = ""
83         Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
84             New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
85         Dim newLineCharProperty As String = comx.NewLine
86         Dim timeOutProperty As Integer = comx.ReadTimeout
87         Dim supressDisplay As Boolean = False
88         Dim iii As Integer = 1
89
90         comx.NewLine = searchChar
91         comx.ReadTimeout = 500
92
```

```vb
93              Do
94                  Try
95                      returnstr = comx.ReadLine()
96                  Catch ex As Exception
97                      'returnstr = ""
98                  End Try
99                  iii += 1
100             Loop Until returnstr.Length > 0 OrElse iii > 16 '8 secs
101
102             If returnstr.Length <= 0 Then 'no response received
103                 returnstr = "No response." & vbLf
104                 supressDisplay = True
105             End If
106
107             'comx.NewLine = searchChar
108             'comx.ReadTimeout = 5000
109
110             'Try
111             '    returnstr = comx.ReadLine()
112             'Catch ex As TimeoutException
113             '    returnstr = "No response." & vbLf
114             '    supressDisplay = True
115             'End Try
116
117             comx.NewLine = newLineCharProperty
118             comx.ReadTimeout = timeOutProperty
119
120             If Not supressDisplay Then
121                 EA.AppendThisString = returnstr '& vbCr
122                 RaiseEvent UpdateOutBoxesEvent(Me, EA)
123             End If
124
125             Return returnstr
126         End Function
127
128         Public Function ReadLine(ByVal searchChar As Char, ByVal timeout As Double) As String
129             'reads until the search char is found, allows for a custom timeout value
130             'faster than ReadUntilChar, but it returns early if no bytes were immediately available to read
131             'also returns nothing if the search character is not found
132
133             Dim returnstr As String = ""
134             Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
135                 New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
136             Dim newLineCharProperty As String = comx.NewLine
137             Dim timeOutProperty As Integer = comx.ReadTimeout
138             Dim supressDisplay As Boolean = False
139             Dim iii As Integer = 1
140
141             comx.NewLine = searchChar
142             comx.ReadTimeout = 500
143
144             Do
145                 Try
146                     returnstr = comx.ReadLine()
147                 Catch ex As Exception
148                     'returnstr = ""
149                 End Try
150                 iii += 1
151             Loop Until returnstr.Length > 0 OrElse iii > CInt(timeout * 2)
152
153             If returnstr.Length <= 0 Then 'no response received
154                 returnstr = "No response." & vbLf
155                 supressDisplay = True
156             End If
157
158             'comx.NewLine = searchChar
159             'comx.ReadTimeout = CInt(timeout * 1000)
160
161             'Try
162             '    returnstr = comx.ReadLine()
163             'Catch ex As TimeoutException
164             '    returnstr = "No response." & vbLf
165             '    supressDisplay = True
166             'End Try
167
168             comx.NewLine = newLineCharProperty
169             comx.ReadTimeout = timeOutProperty
170
171             If Not supressDisplay Then
172                 EA.AppendThisString = returnstr '& vbCr
173                 RaiseEvent UpdateOutBoxesEvent(Me, EA)
174             End If
175
176             Return returnstr
177         End Function
178
179         Public Function ReadLine(ByVal searchChar As Char, ByVal timeout As Integer, _
180         ByRef worker As BackgroundWorker, ByRef e As DoWorkEventArgs) As String
181             'reads until the search char is found, allows for a custom timeout value
182             'exits if a cancel is requested - use this method when waiting for a modem to respond
183
184             Dim count As Integer = 0
```

```vb
185            Dim returnstr As String = ""
186            Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
187                New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
188            Dim newLineCharProperty As String = comx.NewLine
189            Dim timeOutProperty As Integer = comx.ReadTimeout
190
191         comx.NewLine = searchChar
192         comx.ReadTimeout = 1000
193
194         Do
195             Try
196                 returnstr = comx.ReadLine()
197                 Exit Do     'if this statement is reached, there is data to return
198             Catch ex As TimeoutException
199                 count += 1
200                 If worker.CancellationPending Then
201                     Throw New CancelException
202                 End If
203             End Try
204         Loop Until count >= timeout
205
206         comx.NewLine = newLineCharProperty
207         comx.ReadTimeout = timeOutProperty
208
209         EA.AppendThisString = returnstr '& vbCr
210         RaiseEvent UpdateOutBoxesEvent(Me, EA)
211
212         Return returnstr
213
214     End Function
215
216     Public Sub Write(ByVal value As String)
217         'Writes the passed text to the serial port and also to the appropriate window
218         Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
219             New UpdateOutBoxesEventArgs(value, UpdateOutBoxesEventArgs.RW.Write)
220
221         comx.Write(value)
222         RaiseEvent UpdateOutBoxesEvent(Me, EA)
223
224     End Sub
225
226     Public Sub Notify(ByVal value As String)
227         Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
228                 New UpdateOutBoxesEventArgs(value, UpdateOutBoxesEventArgs.RW.Notify)
229
230         RaiseEvent UpdateOutBoxesEvent(Me, EA)
231     End Sub
232
233 #End Region
234
235     Public Delegate Function GetUnitDelegate(ByVal lnum As String) As LDUnit
236     Public Delegate Sub ReturnUnitDelegate(ByVal unit As LDUnit)
237
238     Public Sub Listen(ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs)
239         'listen on one modem for incoming calls
240         'will run on a separate thread, analogous to Dial
241
242         If modemMode = Mode.Dial OrElse worker.CancellationPending Then
243             Exit Sub
244         End If
245
246         Dim unit As New LDUnit("temp")   'Temporary unit object
247         Dim result As String = ""
248         Dim result2 As String = ""
249         Dim uNum As String = ""
250         Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab)}
251         Dim deleg As New GetUnitDelegate(AddressOf Form1.GetUnit)
252         Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
253         Dim newLineChar As String = comx.NewLine
254         Dim ReadTimeOutPropertyValue As Integer = comx.ReadTimeout
255         Dim lockcode As String = "22222222"
256         Dim serial As String = ""
257
258         Try
259             If comx.IsOpen Then
260                 Notify("Port " & comx.PortName & " is in use by another process" & vbLf)
261                 Exit Sub
262                 'comx.Close()
263             End If
264             comx.Open()
265             comx.DiscardInBuffer()  'flush receive buffer
266
267             Notify("*** Monitoring for incoming calls ***" & vbLf & vbLf)
268             Write("AT" & vbCr)
269             result = ReadLine(CChar(vbLf))
270             result &= ReadLine(CChar(vbLf))
271             comx.DiscardInBuffer()
272             Write("ATZ" & vbCr)
273             result = ReadLine(CChar(vbLf))
274             result &= ReadLine(CChar(vbLf))
275             'MessageBox.Show(result, "result after first write")
276             If Not result Like "*OK*" Then
```

```vb
277                     comx.DiscardInBuffer()
278                     Write("+++")
279                     If comx.PortName <> "COM1" Then
280                         ReadLine(CChar(vbLf), 0.5)
281                         ReadLine(CChar(vbLf), 0.5)
282                     End If
283                     Write("ATH" & vbCr)
284                     result = ReadLine(CChar(vbLf))
285                     result &= ReadLine(CChar(vbLf))
286                     Thread.Sleep(1500)
287
288                     comx.DiscardInBuffer()
289                     Write("AT" & vbCr)
290                     result = ReadLine(CChar(vbLf))
291                     result &= ReadLine(CChar(vbLf))
292                     'MessageBox.Show(result, "result after second AT command")
293                     If Not result Like "*OK*" Then
294                         Throw New NoModemException("Modem on " & comx.PortName & " not present or responsive")
295                     End If
296                 End If
297
298             If speakerOn Then
299                 Write("ATM1" & vbCr)
300                 result = ReadLine(CChar(vbLf))
301                 result &= ReadLine(CChar(vbLf))
302             End If
303
304             If Form1.appset.Conn9600 Then
305                 Write("AT+MS=V32,1,2400,9600,2400,9600" & vbCr)
306                 result = ReadLine(CChar(vbLf))
307                 result &= ReadLine(CChar(vbLf))
308             End If
309
310             'wait forever until a "Ring" is received
311             comx.NewLine = vbLf
312             comx.ReadTimeout = 1000
313             Dim speakerold As Boolean = speakerOn
314
315             Do
316                 Try
317                     If worker.CancellationPending Then
318                         Throw New CancelException
319                     End If
320                     If speakerold <> speakerOn Then
321                         If speakerOn Then
322                             Write("ATM1" & vbCr)
323                             result = ReadLine(CChar(vbLf))
324                             result &= ReadLine(CChar(vbLf))
325                         Else
326                             Write("ATM0" & vbCr)
327                             result = ReadLine(CChar(vbLf))
328                             result &= ReadLine(CChar(vbLf))
329                         End If
330                         speakerold = speakerOn
331                     End If
332                     result = comx.ReadLine
333                 Catch ex As TimeoutException
334                     'no more data to read
335                 End Try
336             Loop Until result Like "*RING*"
337
338             comx.ReadTimeout = ReadTimeOutPropertyValue
339             comx.NewLine = newLineChar
340
341             Write("ATA" & vbCr)        'Answer call
342             result = ReadLine(CChar(vbLf), 90, worker, e)
343             result &= ReadLine(CChar(vbLf), 90, worker, e)
344             If worker.CancellationPending Then
345                 Throw New CancelException
346             End If
347             If result Like "*CONNECT*" Then      'get password and unit number
348                 unit.LockCode = lockcode       'only APG units call in
349                 unit = getPass(unit, worker, e)
350                 serial = unit.UnitSerial
351
352                 'get unit number and ask for the appropriate LDUnit object
353                 Write("Q157" & vbCr)
354                 result = ReadLine(CChar(vbLf))
355                 result = result.Trim(trimChars)
356                 If result Like "*ARNING*" Then
357                     Write("Q157" & vbCr)
358                     result = ReadLine(CChar(vbLf))
359                     result = result.Trim(trimChars)
360                 End If
361                 If result Like "*No response.*" Then  'ReadLine timed out
362                     comx.DiscardInBuffer()
363                     Write("Q157" & vbCr)
364                     result = ReadLine(CChar(vbLf))
365                     result = result.Trim(trimChars)
366                     result2 = ReadLine(CChar(vbLf), 1)
367                     result2 = result2.Trim(trimChars)
368                     If Not result Like "*No response.*" Then        'use first read entry
```

```vb
369                        uNum = result
370                    ElseIf Not result2 Like "*No response.*" Then     'use second read entry
371                        uNum = result2
372                    Else
373                        Throw New FormatException("Can't read from unit - will try again later")
374                    End If
375                Else
376                    uNum = result
377                End If
378                If uNum.Length < 2 Then     'This operation assigns an L## LDunit to the unit variable
379                    unit = deleg.Invoke("L0" & uNum)
380                Else
381                    unit = deleg.Invoke("L" & uNum)
382                End If
383
384                If unit.UnitNum = "null" Then
385                    Throw New Exception("Unit is being accessed by another part of the program." & vbLf)
386                End If
387
388                unit.UnitSerial = serial
389                'unit.ResetDataYN = True
390                Download(unit, False, worker, e)
391
392                unit.DialTries = 0
393                unit.DLTries = 0
394                Write("M12" & vbCr)
395                result = ReadLine(CChar(vbLf), 10)
396                comx.DiscardInBuffer()
397                Write("+++")
398                If comx.PortName <> "COM1" Then
399                    ReadLine(CChar(vbLf), 0.5)
400                    ReadLine(CChar(vbLf), 0.5)
401                End If
402                Write("ATH" & vbCr)
403                result = ReadLine(CChar(vbLf))
404                result &= ReadLine(CChar(vbLf))
405
406                'comx.Close() 'moved to Finally block
407
408            Else         'call was dropped
409                Throw New NoConnectException("Unable to connect to remote modem")
410            End If
411
412        Catch ex As NoModemException     'don't use this modem to download the remaining units
413            Notify("Error: " & ex.msg & vbLf)
414            unit.IsDownloadInProgress = False
415            unit.IsDownloadDone = False
416            If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
417                deleg2.Invoke(unit)
418            End If
419            If worker.CancellationPending Then
420                e.Cancel = True
421            End If
422            If comx.IsOpen Then
423                comx.Close()
424            End If
425            Exit Sub
426        Catch ex As NoConnectException       'what happens when the line is busy or unavailable
427            'If comx.IsOpen Then
428            '    comx.Close()
429            'End If
430            Notify("Error: " & ex.msg & vbLf)
431            unit.IsDownloadInProgress = False
432            unit.IsDownloadDone = False
433            unit.DialTries += 1
434            If unit.DialTries >= Form1.appset.DialAttempts Then
435                'stop trying to download unit
436                unit.IsDownloadDone = True
437                unit.DialTries = 0
438                Dim DA As Integer = Form1.appset.DialAttempts + 1
439                Notify("Attempted to dial unit " & unit.UnitNum & " " & DA.ToString & " times." & vbLf)
440                Notify("Will not try again until the next Dial command is issued." & vbLf)
441            End If
442            If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
443                deleg2.Invoke(unit)
444            End If
445        Catch ex As CancelException       'user cancels the operation - close the port and exit
446            Notify(ex.msg & vbLf)
447            unit.IsDownloadInProgress = False
448            unit.IsDownloadDone = False
449            If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
450                deleg2.Invoke(unit)
451            End If
452            e.Cancel = True
453            If comx.IsOpen Then
454                comx.Close()
455            End If
456            Exit Sub
457        Catch ex As InvalidOperationException      'port is closed when attempting to write data
458            Notify("InvalidOperationException" & vbLf)
459            Notify("Error: " & ex.Message & vbLf)
460            'If comx.IsOpen Then
```

```vb
461                    '     comx.Close()
462                    'End If
463                    unit.IsDownloadInProgress = False
464                    unit.IsDownloadDone = False
465                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
466                        deleg2.Invoke(unit)
467                    End If
468            Catch ex As IO.IOException         'port is closed when attempting to write data
469                    Notify("IOException" & vbLf)
470                    Notify("Error: " & ex.Message & vbLf)
471                    'If comx.IsOpen Then
472                    '     comx.Close()
473                    'End If
474                    unit.IsDownloadInProgress = False
475                    unit.IsDownloadDone = False
476                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
477                        deleg2.Invoke(unit)
478                    End If
479            Catch ex As Exception
480                    'will usually be from the download routine, and result from a dropped
481                    'or error-prone connection
482
483                    Write("M12" & vbCr)      'tell 870 to shut down
484                    result = ReadLine(CChar(vbLf), 10)
485                    'result &= ReadLine(CChar(vbLf), 10)
486                    comx.DiscardInBuffer()
487                    Write("+++")
488                    If comx.PortName <> "COM1" Then
489                        ReadLine(CChar(vbLf), 0.5)
490                        ReadLine(CChar(vbLf), 0.5)
491                    End If
492                    Write("ATH" & vbCr)
493                    result = ReadLine(CChar(vbLf))
494                    result &= ReadLine(CChar(vbLf))
495                    'If comx.IsOpen Then
496                    '     comx.Close()
497                    'End If
498                    Notify("Error: " & ex.Message & vbLf)
499                    unit.IsDownloadInProgress = False
500                    unit.IsDownloadDone = False
501                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
502                        deleg2.Invoke(unit)
503                    End If
504
505            Finally
506                    If comx.IsOpen Then
507                        comx.Close()
508                    End If
509            End Try
510
511            If worker.CancellationPending Then
512                    e.Cancel = True
513                    Exit Sub
514            End If
515
516            'conserve memory while the listen routine is recursing
517            ' - the "unit" has been sent back to LDUnitList
518            unit.EList.Clear()
519            unit.IList.Clear()
520            unit.RList.Clear()
521            unit.CList.Clear()
522            unit.QList.Clear()
523            unit.LList.Clear()
524
525            'listen for the next unit to call
526            Listen(worker, e)
527        End Sub
528
529        Public Sub Dial(ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs)
530            'dials a remote modem
531
532            If modemMode = Mode.Listen OrElse worker.CancellationPending Then
533                    Exit Sub
534            End If
535
536            Dim result As String = ""
537            Dim deleg As New GetUnitDelegate(AddressOf Form1.GetUnit)
538            Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
539            Dim unit As LDUnit = deleg.Invoke("-2")
540
541
542            If String.Compare(unit.UnitNum, "null") = 0 Then
543                    Form1.appset.DialInProg = False 'dial finished, stop dialing units
544                    Exit Sub
545            End If
546
547            Notify("*** Now dialing unit " & unit.UnitNum & " at " & Date.Now.ToString("M/d/yyyy HH:mm:ss") _
548            & " ***" & vbLf)
549
550            Try
551                    If unit.IsEnabled = False Then
552                        Throw New NotEnabledException
```

```vb
553                End If
554
555                If comx.IsOpen Then
556                    Notify("Port " & comx.PortName & " is in use by another process" & vbLf)
557                    deleg2.Invoke(unit)
558                    Exit Sub
559                    'comx.Close()
560                End If
561                comx.Open()
562                comx.DiscardInBuffer()   'flush receive buffer
563
564                Write("AT" & vbCr)
565                result = ReadLine(CChar(vbLf))
566                result &= ReadLine(CChar(vbLf))
567                comx.DiscardInBuffer()
568                Write("ATZ" & vbCr)
569                result = ReadLine(CChar(vbLf))
570                result &= ReadLine(CChar(vbLf))
571                'MessageBox.Show(result, "result after first write")
572                If Not result Like "*OK*" Then
573                    comx.DiscardInBuffer()
574                    Write("+++")
575                    If comx.PortName <> "COM1" Then
576                        ReadLine(CChar(vbLf), 0.5)
577                        ReadLine(CChar(vbLf), 0.5)
578                    End If
579                    Write("ATH" & vbCr)
580                    result = ReadLine(CChar(vbLf))
581                    result &= ReadLine(CChar(vbLf))
582                    Thread.Sleep(1500)
583
584                    comx.DiscardInBuffer()
585                    Write("AT" & vbCr)
586                    result = ReadLine(CChar(vbLf))
587                    result &= ReadLine(CChar(vbLf))
588                    'MessageBox.Show(result, "result after second AT command")
589                    If Not result Like "*OK*" Then
590                        Throw New NoModemException("Modem on " & comx.PortName & " not present or responsive")
591                    End If
592                End If
593                If speakerOn Then
594                    Write("ATM1" & vbCr)
595                    result = ReadLine(CChar(vbLf))
596                    result &= ReadLine(CChar(vbLf))
597                End If
598                If Form1.appset.Conn9600 Then
599                    Write("AT+MS=V32,1,2400,9600,2400,9600" & vbCr)
600                    result = ReadLine(CChar(vbLf))
601                    result &= ReadLine(CChar(vbLf))
602                End If
603
604                'If unit.UnitOwner = LDUnit.Owner.CERL Then   'limit modem speed to 14400 bps
605                '    If comx.PortName = "COM1" Then
606                '        Write("AT$MB14400 \N3" & vbCr)
607                '        result = ReadLine(CChar(vbLf))
608                '        result &= ReadLine(CChar(vbLf))
609                '    Else
610                '        Write("AT+MS=V32B,1,300,14400,300,14400" & vbCr)
611                '        result = ReadLine(CChar(vbLf))
612                '        result &= ReadLine(CChar(vbLf))
613                '        Write("AT\N3" & vbCr)
614                '        result = ReadLine(CChar(vbLf))
615                '        result &= ReadLine(CChar(vbLf))
616                '    End If
617                'End If
618
619                Write("ATDT" & unit.UnitPhoneNum & vbCr)
620                result = ReadLine(CChar(vbLf), 90, worker, e)
621                result &= ReadLine(CChar(vbLf), 90, worker, e)
622                If worker.CancellationPending Then
623                    Throw New CancelException
624                End If
625                If result Like "*OK*" Then
626                    result &= ReadLine(CChar(vbLf), 90, worker, e)
627                    If worker.CancellationPending Then
628                        Throw New CancelException
629                    End If
630                End If
631                If Not result Like "*CONNECT*" Then
632                    comx.DiscardInBuffer()
633                    Write("ATZ" & vbCr)
634                    result = ReadLine(CChar(vbLf))
635                    result &= ReadLine(CChar(vbLf))
636                    If speakerOn Then
637                        Write("ATM1" & vbCr)
638                        result = ReadLine(CChar(vbLf))
639                        result &= ReadLine(CChar(vbLf))
640                    End If
641                    If Form1.appset.Conn9600 Then
642                        Write("AT+MS=V32,1,2400,9600,2400,9600" & vbCr)
643                        result = ReadLine(CChar(vbLf))
644                        result &= ReadLine(CChar(vbLf))
```

```vb
645                     End If
646
647                     Write("ATDT" & unit.UnitPhoneNum & vbCr)
648                     result = ReadLine(CChar(vbLf), 90, worker, e)
649                     result &= ReadLine(CChar(vbLf), 90, worker, e)
650                     If worker.CancellationPending Then
651                         Throw New CancelException
652                     End If
653                     If result Like "*OK*" Then
654                         result &= ReadLine(CChar(vbLf), 90, worker, e)
655                         If worker.CancellationPending Then
656                             Throw New CancelException
657                         End If
658                     End If
659                     If Not result Like "*CONNECT*" Then
660                         Throw New NoConnectException("Unable to connect to remote modem")
661                     End If
662                 End If
663
664                 If unit.Initialize Then
665                     ReconfigureUnit(unit, True, worker, e)
666                 Else
667                     Download(unit, True, worker, e)  'executes download on same thread
668                 End If
669
670                 comx.DiscardInBuffer()
671                 Write("M12" & vbCr)
672                 result = ReadLine(CChar(vbLf), 10)
673                 'result &= ReadLine(CChar(vbLf), 10)
674                 comx.DiscardInBuffer()
675                 Write("+++")
676                 If comx.PortName <> "COM1" Then
677                     ReadLine(CChar(vbLf), 0.5)
678                     ReadLine(CChar(vbLf), 0.5)
679                 End If
680                 Write("ATH" & vbCr)
681                 result = ReadLine(CChar(vbLf))
682                 result &= ReadLine(CChar(vbLf))
683
684
685                 'comx.Close() ' moved to Finally block
686
687             Catch ex As NoModemException    'don't use this modem to download the remaining units
688                 Notify("Error: " & ex.msg & vbLf)
689                 unit.IsDownloadInProgress = False
690                 unit.IsDownloadDone = False
691                 If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
692                     deleg2.Invoke(unit)
693                 End If
694                 If worker.CancellationPending Then
695                     e.Cancel = True
696                 End If
697                 If comx.IsOpen Then
698                     comx.Close()
699                 End If
700                 Exit Sub
701             Catch ex As NoConnectException       'what happens when the line is busy or unavailable
702                 'If comx.IsOpen Then
703                 '    comx.Close()
704                 'End If
705                 Notify("Error: " & ex.msg & vbLf)
706                 unit.IsDownloadInProgress = False
707                 unit.IsDownloadDone = False
708                 unit.DialTries += 1
709                 If unit.DialTries >= Form1.appset.DialAttempts Then
710                     'stop trying to download unit
711                     unit.IsDownloadDone = True
712                     unit.DialTries = 0
713                     Dim DA As Integer = Form1.appset.DialAttempts + 1
714                     Notify("Attempted to dial unit " & unit.UnitNum & " " & DA.ToString & " times." & vbLf)
715                     Notify("Will not try again until the next Dial command is issued." & vbLf)
716                 End If
717                 If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
718                     deleg2.Invoke(unit)
719                 End If
720             Catch ex As NotEnabledException
721                 Notify("Skipping unit " & unit.UnitNum & " because is not enabled." & vbLf)
722                 Dim EA As New StatusBarEventArgs(unit.UnitNum, StatusBarEventArgs.SBAction.Remove)
723                 'RaiseEvent UpdateStatusBar(Me, EA)
724                 RaiseEvent SBUnitCounter(Me, EA)
725                 RaiseEvent UpdateMessages(Me, EA)
726
727                 'If comx.IsOpen Then
728                 '    comx.Close()
729                 'End If
730                 unit.IsDownloadInProgress = False
731                 unit.IsDownloadDone = True
732                 If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
733                     deleg2.Invoke(unit)
734                 End If
735             Catch ex As CancelException     'user cancels the operation - close the port and exit
736                 Notify(ex.msg & vbLf)
```

```vb
737                    unit.IsDownloadInProgress = False
738                    unit.IsDownloadDone = False
739                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
740                        deleg2.Invoke(unit)
741                    End If
742                    e.Cancel = True
743                    If comx.IsOpen Then
744                        comx.Close()
745                    End If
746                    Exit Sub
747                Catch ex As InvalidOperationException       'port is closed when attempting to write data
748                    Notify("InvalidOperationException" & vbLf)
749                    Notify("Error: " & ex.Message & vbLf)
750                    unit.IsDownloadInProgress = False
751                    unit.IsDownloadDone = False
752                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
753                        deleg2.Invoke(unit)
754                    End If
755                    'If comx.IsOpen Then
756                    '    comx.Close()
757                    'End If
758                Catch ex As IO.IOException       'port is closed when attempting to write data
759                    Notify("IOException" & vbLf)
760                    Notify("Error: " & ex.Message & vbLf)
761                    unit.IsDownloadInProgress = False
762                    unit.IsDownloadDone = False
763                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
764                        deleg2.Invoke(unit)
765                    End If
766                    'If comx.IsOpen Then
767                    '    comx.Close()
768                    'End If
769                Catch ex As Exception
770                    'will usually be from the download routine, and result from a dropped
771                    'or error-prone connection
772
773                    Write("M12" & vbCr)      'tell 870 to shut down
774                    result = ReadLine(CChar(vbLf), 10)
775                    'result &= ReadLine(CChar(vbLf), 10)
776                    Write("+++")
777                    If comx.PortName <> "COM1" Then
778                        ReadLine(CChar(vbLf), 0.5)
779                        ReadLine(CChar(vbLf), 0.5)
780                    End If
781                    Write("ATH" & vbCr)
782                    result = ReadLine(CChar(vbLf))
783                    result &= ReadLine(CChar(vbLf))
784                    Notify("Error: " & ex.Message & vbLf)
785                    unit.IsDownloadInProgress = False
786                    unit.IsDownloadDone = False
787                    unit.DialTries += 1
788                    If unit.DialTries > Form1.appset.DialAttempts Then
789                        'stop trying to download unit
790                        unit.IsDownloadDone = True
791                        Dim DA As Integer = Form1.appset.DialAttempts + 1
792                        Notify("Attempted to dial unit " & unit.UnitNum & " " & DA.ToString & " times." & vbLf)
793                        Notify("Will not try again until the next Dial command is issued." & vbLf)
794                    End If
795                    If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
796                        deleg2.Invoke(unit)
797                    End If
798                Finally
799                    If comx.IsOpen Then
800                        comx.Close()
801                    End If
802                End Try
803
804                If worker.CancellationPending Then
805                    e.Cancel = True
806                    Exit Sub
807                End If
808
809                'conserve memory while the listen routine is recursing
810                ' - the "unit" has been sent back to LDUnitList
811                unit.EList.Clear()
812                unit.IList.Clear()
813                unit.RList.Clear()
814                unit.CList.Clear()
815                unit.QList.Clear()
816                unit.LList.Clear()
817
818                'repeat until all units are done
819                Dial(worker, e)
820        End Sub
821
822        Public Sub td(ByVal unit As LDUnit, ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs)
823                'perform some routine on all units
824                Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
825
826                Dim result As String = ""
827                Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab)}
828
```

```vb
829
830          unit = getPass(unit, worker, e)
831
832          Write("R93" & vbCr)
833          result = ReadLine(CChar(vbLf))
834          result = result.Trim(trimChars)
835          If result Like "*@*" Then          'turn off L-D error checking
836              Write("S205,0F" & vbCr)
837              result = ReadLine(CChar(vbLf))
838              Write("R93" & vbCr)
839              result = ReadLine(CChar(vbLf))
840              result = result.Trim(trimChars)
841          ElseIf result Like "*ARNING*" Then
842              Write("R93" & vbCr)
843              result = ReadLine(CChar(vbLf))
844              result = result.Trim(trimChars)
845          End If
846
847          If worker.CancellationPending Then
848              e.Cancel = True
849              Throw New CancelException
850          End If
851
852          Write("Q158" & vbCr)
853          result = ReadLine(CChar(vbLf))
854
855          If worker.CancellationPending Then
856              Write("M12" & vbCr)
857              Throw New CancelException
858          End If
859
860          Write("S158,X4 E0 Q0 V0 T M1 S0=5 &D3" & vbCr)
861          result = ReadLine(CChar(vbLf))
862          Write("Q158" & vbCr)
863          result = ReadLine(CChar(vbLf))
864
865      End Sub
866
867      Public Sub Download(ByVal unit As LDUnit, ByVal askPass As Boolean, _
868      ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs)
869          'Initiates a data transfer from the 870
870          'This procedure is common to both the dial out and dial in routines
871          'This procedure assumes the host computer is connected to the 870
872
873          Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
874
875          Dim result As String = ""
876          Dim result2 As String = ""
877          Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab)}
878          Dim parseRes As Boolean = True
879          Dim parseRes2 As Boolean = True
880          Dim excdThreshTemp As Double = -1
881          Dim calLevel As Double = -1
882
883          Dim numExceedances As Integer = -1   'r93, E records
884          Dim numIntervals As Integer = -1     'r94, I records
885          Dim numStartStops As Integer = -1    'r92, L records
886          Dim numCalibrations As Integer = -1  'r97, C records
887          Dim numreportlines As Integer = 0
888          Dim battVoltage As Double = -1.1      'r86
889          Dim unitErrorString As String = ""    'r98
890          Dim lotsOfExcds As Boolean = False
891          Dim lotsOfIntvs As Boolean = False
892          Dim report As List(Of String) = New List(Of String)     'the report, each line is a List entry
893          Dim EA2 As UpdateOutBoxesEventArgs = _
894                      New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
895
896
897          unit.IsDownloadDone = False
898          unit.IsDownloadInProgress = False
899          If Not comx.IsOpen Then
900              deleg2.Invoke(unit)
901              Exit Sub
902          End If
903
904          unit.IsDownloadInProgress = True
905
906          'reset report data
907          unit.NumCalibrations = -1
908          unit.NumExceedances = -1
909          unit.NumIntervals = -1
910          unit.NumStartStops = -1
911          unit.ErrorString = ""
912          unit.BattVoltage = -1
913          unit.EList.Clear()
914          unit.IList.Clear()
915          unit.RList.Clear()
916          unit.QList.Clear()
917          unit.LList.Clear()
918          unit.CList.Clear()
919
920          'retrieve password, cancel download if it is in progress
```

```vb
921         If askPass Then
922             unit = getPass(unit, worker, e)
923         End If
924
925         'retrieve device status
926         'to put read data into local variables, trim the cr, lf, : and spaces from the string,
927         'then parse it to an integer or double
928
929         'RemoveHandler comx.DataReceived, AddressOf ReadPort
930         Notify("*** Downloading unit " & unit.UnitNum & " at " & Date.Now.ToString("M/d/yyyy HH:mm:ss") _
931         & " ***" & vbLf)
932
933         Write("R93" & vbCr)
934         result = ReadLine(CChar(vbLf))
935         result = result.Trim(trimChars)
936         If result Like "*@*" Then        'turn off L-D error checking
937             Write("S205,0F" & vbCr)
938             result = ReadLine(CChar(vbLf))
939             Write("R93" & vbCr)
940             result = ReadLine(CChar(vbLf))
941             result = result.Trim(trimChars)
942         ElseIf result Like "*ARNING*" Then
943             Write("R93" & vbCr)
944             result = ReadLine(CChar(vbLf))
945             result = result.Trim(trimChars)
946         End If
947         If result Like "*No response.*" Then  'ReadLine timed out
948             Dim tempint, tempint2 As Integer
949
950             comx.DiscardInBuffer()
951             Write("R93" & vbCr)
952             result = ReadLine(CChar(vbLf))
953             result = result.Trim(trimChars)
954             result2 = ReadLine(CChar(vbLf), 1)
955             result2 = result2.Trim(trimChars)
956             parseRes = Integer.TryParse(result, tempint)
957             parseRes2 = Integer.TryParse(result2, tempint2)
958             If parseRes2 Then         'use second read entry
959                 numExceedances = tempint2
960             ElseIf parseRes Then      'use first read entry
961                 numExceedances = tempint
962             Else
963                 Throw New FormatException("Can't read from unit - will try again later")
964             End If
965         Else
966             numExceedances = Integer.Parse(result)
967         End If
968
969         If numExceedances > 50 Then
970             lotsOfExcds = True
971         End If
972
973
974         Write("R94" & vbCr)
975         result = ReadLine(CChar(vbLf))
976         result = result.Trim(trimChars)
977         If result Like "*ARNING*" Then
978             Write("R94" & vbCr)
979             result = ReadLine(CChar(vbLf))
980             result = result.Trim(trimChars)
981         End If
982         If result Like "*No response.*" Then  'ReadLine timed out
983             Dim tempint, tempint2 As Integer
984
985             comx.DiscardInBuffer()
986             Write("R94" & vbCr)
987             result = ReadLine(CChar(vbLf))
988             result = result.Trim(trimChars)
989             result2 = ReadLine(CChar(vbLf), 1)
990             result2 = result2.Trim(trimChars)
991             parseRes = Integer.TryParse(result, tempint)
992             parseRes2 = Integer.TryParse(result2, tempint2)
993             If parseRes2 Then         'use second read entry
994                 numIntervals = tempint2
995             ElseIf parseRes Then      'use first read entry
996                 numIntervals = tempint
997             Else
998                 Throw New FormatException("Can't read from unit - will try again later")
999             End If
1000        Else
1001            numIntervals = Integer.Parse(result)
1002        End If
1003
1004        If numIntervals > 50 Then
1005            lotsofintvs = True
1006        End If
1007
1008
1009        Write("R92" & vbCr)
1010        result = ReadLine(CChar(vbLf))
1011        result = result.Trim(trimChars)
1012        If result Like "*ARNING*" Then
```

```vb
1013                Write("R92" & vbCr)
1014                result = ReadLine(CChar(vbLf))
1015                result = result.Trim(trimChars)
1016            End If
1017            If result Like "*No response.*" Then   'ReadLine timed out
1018                Dim tempint, tempint2 As Integer
1019
1020                comx.DiscardInBuffer()
1021                Write("R92" & vbCr)
1022                result = ReadLine(CChar(vbLf))
1023                result = result.Trim(trimChars)
1024                result2 = ReadLine(CChar(vbLf), 1)
1025                result2 = result2.Trim(trimChars)
1026                parseRes = Integer.TryParse(result, tempint)
1027                parseRes2 = Integer.TryParse(result2, tempint2)
1028                If parseRes2 Then          'use second read entry
1029                    numStartStops = tempint2
1030                ElseIf parseRes Then     'use first read entry
1031                    numStartStops = tempint
1032                Else
1033                    Throw New FormatException("Can't read from unit - will try again later")
1034                End If
1035            Else
1036                numStartStops = Integer.Parse(result)
1037            End If
1038
1039
1040            Write("R97" & vbCr)
1041            result = ReadLine(CChar(vbLf))
1042            result = result.Trim(trimChars)
1043            If result Like "*ARNING*" Then
1044                Write("R97" & vbCr)
1045                result = ReadLine(CChar(vbLf))
1046                result = result.Trim(trimChars)
1047            End If
1048            If result Like "*No response.*" Then   'ReadLine timed out
1049                Dim tempint, tempint2 As Integer
1050
1051                comx.DiscardInBuffer()
1052                Write("R97" & vbCr)
1053                result = ReadLine(CChar(vbLf))
1054                result = result.Trim(trimChars)
1055                result2 = ReadLine(CChar(vbLf), 1)
1056                result2 = result2.Trim(trimChars)
1057                parseRes = Integer.TryParse(result, tempint)
1058                parseRes2 = Integer.TryParse(result2, tempint2)
1059                If parseRes2 Then        'use second read entry
1060                    numCalibrations = tempint2
1061                ElseIf parseRes Then     'use first read entry
1062                    numCalibrations = tempint
1063                Else
1064                    Throw New FormatException("Can't read from unit - will try again later")
1065                End If
1066            Else
1067                numCalibrations = Integer.Parse(result)
1068            End If
1069
1070
1071            Dim trimchars2() As Char = {"V"c}
1072            Write("R86" & vbCr)
1073            result = ReadLine(CChar(vbLf))
1074            result = result.Trim(trimChars)
1075            result = result.Trim(trimchars2)
1076            If result Like "*ARNING*" Then
1077                Write("R86" & vbCr)
1078                result = ReadLine(CChar(vbLf))
1079                result = result.Trim(trimChars)
1080                result = result.Trim(trimchars2)
1081            End If
1082            If result Like "*No response.*" Then   'ReadLine timed out
1083                Dim tempd, tempd2 As Double
1084
1085                comx.DiscardInBuffer()
1086                Write("R86" & vbCr)
1087                result = ReadLine(CChar(vbLf))
1088                result = result.Trim(trimChars)
1089                result = result.Trim(trimchars2)
1090                result2 = ReadLine(CChar(vbLf), 1)
1091                result2 = result2.Trim(trimChars)
1092                result2 = result2.Trim(trimchars2)
1093                parseRes = Double.TryParse(result, tempd)
1094                parseRes2 = Double.TryParse(result2, tempd2)
1095                If parseRes2 Then          'use second read entry
1096                    battVoltage = tempd2
1097                ElseIf parseRes Then     'use first read entry
1098                    battVoltage = tempd
1099                Else
1100                    Throw New FormatException("Can't read from unit - will try again later")
1101                End If
1102            Else
1103                battVoltage = Double.Parse(result)
1104            End If
```

```vb
1105            If battVoltage > 20 Then
1106                battVoltage *= 0.1
1107            End If
1108
1109            Write("R98" & vbCr)
1110            result = ReadLine(CChar(vbLf))
1111            result = result.Trim(trimChars)
1112            If result Like "*ARNING*" Then
1113                Write("R98" & vbCr)
1114                result = ReadLine(CChar(vbLf))
1115                result = result.Trim(trimChars)
1116            End If
1117            If result Like "*No response.*" Then   'ReadLine timed out
1118                comx.DiscardInBuffer()
1119                Write("R98" & vbCr)
1120                result = ReadLine(CChar(vbLf))
1121                result = result.Trim(trimChars)
1122                result2 = ReadLine(CChar(vbLf), 1)
1123                result2 = result2.Trim(trimChars)
1124                If Not result Like "*No response.*" Then        'use first read entry
1125                    unitErrorString = result
1126                ElseIf Not result2 Like "*No response.*" Then    'use second read entry
1127                    unitErrorString = result2
1128                Else
1129                    Throw New FormatException("Can't read from unit - will try again later")
1130                End If
1131            Else
1132                unitErrorString = result
1133            End If
1134
1135            'subtract 128 from each warning code to make codes easier to read
1136            'Actual errors (rare) are numbered 1 thru 8 - add 100 to them to differentiate from the
1137            'more common warning codes
1138            Dim trimChars3() As Char = {",","c"}
1139            Dim esTemp() As String = unitErrorString.Split(trimChars3, StringSplitOptions.RemoveEmptyEntries)
1140            Dim errorInts(esTemp.Length - 1) As Integer
1141
1142            unitErrorString = ""
1143
1144            For i As Integer = 0 To esTemp.Length - 1
1145                Dim tryp As Boolean = Integer.TryParse(esTemp(i), errorInts(i))
1146                If tryp Then 'add or subtract and append result to the error string
1147                    If errorInts(i) > 127 Then
1148                        errorInts(i) -= 128 'warning codes now numbered 1-43
1149                    Else
1150                        errorInts(i) += 100 'error codes now numbered 101-108
1151                    End If
1152                    If i < esTemp.Length - 1 Then
1153                        unitErrorString &= errorInts(i).ToString & ","
1154                    Else
1155                        unitErrorString &= errorInts(i).ToString
1156                    End If
1157                    'End If
1158                End If
1159            Next
1160
1161
1162
1163            'get cal level
1164            Write("R107" & vbCr)
1165            result = ReadLine(CChar(vbLf))
1166            result = result.Trim(trimChars)
1167            If result Like "*ARNING*" OrElse result.Length > 7 Then
1168                Write("R107" & vbCr)
1169                result = ReadLine(CChar(vbLf))
1170                result = result.Trim(trimChars)
1171            End If
1172            If result Like "*No response.*" Then   'ReadLine timed out
1173                Dim tempd, tempd2 As Double
1174
1175                comx.DiscardInBuffer()
1176                Write("R107" & vbCr)
1177                result = ReadLine(CChar(vbLf))
1178                result = result.Trim(trimChars)
1179                result2 = ReadLine(CChar(vbLf), 1)
1180                result2 = result2.Trim(trimChars)
1181                parseRes = Double.TryParse(result, tempd)
1182                parseRes2 = Double.TryParse(result2, tempd2)
1183                If parseRes2 Then           'use second read entry
1184                    calLevel = tempd2
1185                ElseIf parseRes Then        'use first read entry
1186                    calLevel = tempd
1187                Else
1188                    Throw New FormatException("Can't read from unit - will try again later")
1189                End If
1190            Else
1191                calLevel = Double.Parse(result)
1192            End If
1193
1194
1195            'MessageBox.Show(numExceedances.ToString & vbCr & numIntervals.ToString & vbCr & numStartStops. ↵
          ToString & vbCr & result & vbCr & battVoltage)
```

```vb
1196
1197            If battVoltage < 0 OrElse numStartStops < 0 OrElse numIntervals < 0 OrElse numExceedances < 0 Then
1198                unit.IsDownloadInProgress = False
1199                deleg2.Invoke(unit)
1200                Exit Sub
1201            End If
1202
1203
1204
1205            'make sure all units are in battery cut-off mode, since they run on solar power
1206            'and will require a battery change if they are left on too long
1207
1208            Write("Q159" & vbCr)
1209            result = ReadLine(CChar(vbLf))
1210            result = result.Trim(trimChars)
1211            If Not result Like "*Ext*" Then
1212                Write("S159,1" & vbCr)
1213                result = ReadLine(CChar(vbLf))
1214                Write("S159,1" & vbCr)
1215                result = ReadLine(CChar(vbLf))
1216            End If
1217
1218
1219
1220            'check exceedance threshold
1221            If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
1222                Write("Q64" & vbCr)
1223                result = ReadLine(CChar(vbLf))
1224                result = result.Trim(trimChars)
1225                If result Like "*ARNING*" Then
1226                    Write("Q64" & vbCr)
1227                    result = ReadLine(CChar(vbLf))
1228                    result = result.Trim(trimChars)
1229                End If
1230                If result Like "*No response.*" Then  'ReadLine timed out
1231                    Dim tempd, tempd2 As Double
1232
1233                    comx.DiscardInBuffer()
1234                    Write("Q64" & vbCr)
1235                    result = ReadLine(CChar(vbLf))
1236                    result = result.Trim(trimChars)
1237                    result2 = ReadLine(CChar(vbLf), 1)
1238                    result2 = result2.Trim(trimChars)
1239                    parseRes = Double.TryParse(result, tempd)
1240                    parseRes2 = Double.TryParse(result2, tempd2)
1241                    If parseRes2 Then       'use second read entry
1242                        excdThreshTemp = tempd2
1243                    ElseIf parseRes Then    'use first read entry
1244                        excdThreshTemp = tempd
1245                    Else
1246                        Throw New FormatException("Can't read from unit - will try again later")
1247                    End If
1248                Else
1249                    excdThreshTemp = Double.Parse(result)
1250                End If
1251            Else
1252                Write("Q64" & vbCr)
1253                result = ReadLine(CChar(vbLf))
1254                result = result.Trim(trimChars)
1255                If result Like "*ARNING*" Then
1256                    Write("Q64" & vbCr)
1257                    result = ReadLine(CChar(vbLf))
1258                    result = result.Trim(trimChars)
1259                End If
1260                If result Like "*No response.*" Then  'ReadLine timed out
1261                    Dim tempd, tempd2 As Double
1262
1263                    comx.DiscardInBuffer()
1264                    Write("Q64" & vbCr)
1265                    result = ReadLine(CChar(vbLf))
1266                    result = result.Trim(trimChars)
1267                    result2 = ReadLine(CChar(vbLf), 1)
1268                    result2 = result2.Trim(trimChars)
1269                    parseRes = Double.TryParse(result, tempd)
1270                    parseRes2 = Double.TryParse(result2, tempd2)
1271                    If parseRes2 Then       'use second read entry
1272                        excdThreshTemp = tempd2
1273                    ElseIf parseRes Then    'use first read entry
1274                        excdThreshTemp = tempd
1275                    Else
1276                        Throw New FormatException("Can't read from unit - will try again later")
1277                    End If
1278                Else
1279                    excdThreshTemp = Double.Parse(result)
1280                End If
1281            End If
1282
1283
1284
1285            'Get the unit time and the PC time to check the time difference
1286
1287            Dim unitDateAvg As Date
```

```vb
1288            Dim hostDateAvg As Date
1289
1290            Dim RTcurrent As ReferenceTime = GetUnitTime(worker, e)
1291            unitDateAvg = RTcurrent.UnitTimeZero
1292            hostDateAvg = RTcurrent.HostTimeZero
1293
1294            'MessageBox.Show(unitDateAvg & vbLf & hostDateAvg, "Unit Time, Host Time")
1295
1296            'Rounding statement - don't convert offset to an integer until it's time to paste the value into the ↙
        report
1297            'tOffsetAvg1 = CInt(Math.Round(tOffsetAvg1, 0, MidpointRounding.AwayFromZero))
1298
1299            'for testing only
1300            'unit.IncludeR = True
1301            'unit.IncludeQ = True
1302            'unit.IncludeLC = True
1303            'for testing only
1304
1305
1306
1307            'Download the report, per the parameters set in the unit object
1308            Write("P10" & vbCr)
1309            result = ReadLine(CChar(vbLf))
1310            Write("P10" & vbCr)
1311            result = ReadLine(CChar(vbLf))
1312            Write("P10" & vbCr)
1313            result = ReadLine(CChar(vbLf))
1314            If Not unit.IncludeR Then
1315                Write("S177,0" & vbCr)
1316                result = ReadLine(CChar(vbLf))
1317                Write("S177,0" & vbCr)
1318                result = ReadLine(CChar(vbLf))
1319                result = result.Trim(trimChars)
1320                If Not result Like "" Then      'send command again
1321                    Write("S177,0" & vbCr)
1322                    result = ReadLine(CChar(vbLf))
1323                End If
1324            Else
1325                numreportlines += 220
1326            End If
1327            If Not unit.IncludeLC Then
1328                Write("S178,0" & vbCr)
1329                result = ReadLine(CChar(vbLf))
1330                Write("S178,0" & vbCr)
1331                result = ReadLine(CChar(vbLf))
1332                result = result.Trim(trimChars)
1333                If Not result Like "" Then      'send command again
1334                    Write("S178,0" & vbCr)
1335                    result = ReadLine(CChar(vbLf))
1336                End If
1337            Else
1338                numreportlines += numStartStops
1339                numreportlines += numCalibrations
1340            End If
1341            If Not unit.IncludeQ Then
1342                Write("S179,0" & vbCr)
1343                result = ReadLine(CChar(vbLf))
1344                Write("S179,0" & vbCr)
1345                result = ReadLine(CChar(vbLf))
1346                result = result.Trim(trimChars)
1347                If Not result Like "" Then      'send command again
1348                    Write("S179,0" & vbCr)
1349                    result = ReadLine(CChar(vbLf))
1350                End If
1351            Else
1352                numreportlines += 231
1353            End If
1354
1355            numreportlines += numExceedances
1356            numreportlines += numIntervals
1357
1358            'MessageBox.Show(numReportLines.ToString)
1359
1360            Dim RQCounter As Integer = 0
1361            Dim ECounter As Integer = 0
1362            Dim ICounter As Integer = 0
1363            Dim newLineChar As String = comx.NewLine
1364            Dim ReadTimeOutPropertyValue As Integer = comx.ReadTimeout
1365            comx.NewLine = vbLf
1366            comx.ReadTimeout = 4000
1367
1368            Try
1369                Write("P100" & vbCr)
1370                For i As Integer = 1 To numreportlines + 2000
1371                    If worker.CancellationPending Then
1372                        e.Cancel = True
1373                        Write("M12" & vbCr)
1374                        result = ReadLine(CChar(vbLf))
1375                        Throw New CancelException
1376                    End If
1377                    report.Add(comx.ReadLine)
1378                    If (IsR(report.Item(i - 1)) OrElse IsQ(report.Item(i - 1)) _
```

```vb
1379                         OrElse IsL(report.Item(i - 1)) OrElse IsC(report.Item(i - 1))) Then
1380                             If RQCounter < 80 Then
1381                                 EA2.AppendThisString = "."
1382                                 RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1383                                 RQCounter += 1
1384                             Else
1385                                 EA2.AppendThisString = vbLf & "."
1386                                 RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1387                                 RQCounter = 1
1388                             End If
1389                         End If
1390                         If IsE(report.Item(i - 1)) Then
1391                             If lotsOfExcds Then   'don't print each excd record to screen
1392                                 ECounter += 1
1393                                 If ECounter = 1 Then
1394                                     EA2.AppendThisString = "There are " & numExceedances.ToString & _
1395                                     " exceedance records on unit." & vbLf & _
1396                                     "Reading Exceedances... "
1397                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1398                                 ElseIf ECounter Mod 5 = 0 Then
1399                                     EA2.AppendThisString = "."
1400                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1401                                     If ECounter Mod 50 = 0 Then
1402                                         EA2.AppendThisString = vbLf & ECounter.ToString & " of " & _
1403                                         numExceedances.ToString & " exceedances read"
1404                                         RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1405                                     End If
1406                                 ElseIf ECounter >= numExceedances Then
1407                                     EA2.AppendThisString = vbLf & "All exceedances read." & vbLf
1408                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1409                                 End If
1410
1411                             Else
1412                                 EA2.AppendThisString = report.Item(i - 1) '& vbLf
1413                                 RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1414                             End If
1415                         End If
1416                         If IsI(report.Item(i - 1)) Then
1417                             If lotsOfIntvs Then   'don't print each interval record to screen
1418                                 ICounter += 1
1419                                 If ICounter = 1 Then
1420                                     EA2.AppendThisString = "There are " & numIntervals.ToString & _
1421                                     " interval records on unit." & vbLf & _
1422                                     "Reading Intervals... "
1423                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1424                                 ElseIf ICounter Mod 5 = 0 Then
1425                                     EA2.AppendThisString = "."
1426                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1427                                     If ICounter Mod 50 = 0 Then
1428                                         EA2.AppendThisString = vbLf & ICounter.ToString & " of " & _
1429                                         numIntervals.ToString & " intervals read"
1430                                         RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1431                                     End If
1432                                 ElseIf ICounter >= numIntervals Then
1433                                     EA2.AppendThisString = vbLf & "All intervals read." & vbLf
1434                                     RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1435                                 End If
1436
1437                             Else
1438                                 EA2.AppendThisString = report.Item(i - 1) '& vbLf
1439                                 RaiseEvent UpdateOutBoxesEvent(Me, EA2)
1440                             End If
1441                         End If
1442                     Next
1443             Catch ex As TimeoutException
1444                 'no more data to read
1445             End Try
1446
1447             comx.ReadTimeout = ReadTimeOutPropertyValue
1448             comx.NewLine = newLineChar
1449
1450             'MessageBox.Show(unitDateAvg.ToString & vbCr & hostDateAvg.ToString)
1451
1452
1453
1454             'Check format of the report, if all is well proceed, otherwise download the report again
1455
1456             Dim EList As New List(Of String)
1457             Dim IList As New List(Of String)
1458             Dim LList As New List(Of String)
1459             Dim CList As New List(Of String)
1460             Dim RList As New List(Of String)
1461             Dim QList As New List(Of String)
1462             Dim downloadAgain As Boolean = False
1463             Dim IsBadRecords As Boolean = False
1464
1465             'MessageBox.Show(report.Item(0) & vbCr & report.Item(0).Length.ToString)
1466
1467             If report.Count < numreportlines Then        'repeat download immediately if DL is clearly not
1468     complete
1468                 Write("P999" & vbCr)
1469                 Thread.Sleep(1500)
```

```vb
1470                    comx.DiscardInBuffer()
1471                    Write("P999" & vbCr)
1472                    Thread.Sleep(1500)
1473                    comx.DiscardInBuffer()
1474                    unit.DLTries += 1
1475                    If unit.DLTries < 4 Then
1476                        Notify("Report is incomplete - downloading again..." & vbLf)
1477                        Download(unit, False, worker, e)
1478                        Exit Sub
1479                    Else
1480                        unit.IsDownloadInProgress = False
1481                        unit.IsDownloadDone = False
1482                        unit.DLTries = 0
1483                        Notify("Warning! Unit " & unit.UnitNum & " has a bad connection" & vbLf)
1484                        Notify("   and cannot transfer its data." & vbLf)
1485                        Notify("Download will resume after the other units have finished." & vbLf)
1486                        deleg2.Invoke(unit)
1487                        Exit Sub
1488                    End If
1489                End If
1490
1491            'Copy valid entries to their respective list
1492            EList = report.FindAll(AddressOf IsE)
1493            IList = report.FindAll(AddressOf IsI)
1494
1495            If EList.Count < numExceedances OrElse IList.Count < numIntervals Then
1496                IsBadRecords = True
1497            End If
1498
1499            If unit.IncludeR Then
1500                RList = report.FindAll(AddressOf IsR)
1501                If RList.Count < 220 Then
1502                    IsBadRecords = True
1503                End If
1504            End If
1505            If unit.IncludeQ Then
1506                QList = report.FindAll(AddressOf IsQ)
1507                If QList.Count < 231 Then
1508                    IsBadRecords = True
1509                End If
1510            End If
1511            If unit.IncludeLC Then
1512                LList = report.FindAll(AddressOf IsL)
1513                CList = report.FindAll(AddressOf IsC)
1514                If LList.Count < numStartStops OrElse CList.Count < numCalibrations Then
1515                    IsBadRecords = True
1516                End If
1517            End If
1518
1519            'MessageBox.Show(EList.Count.ToString & vbCr & IList.Count.ToString & vbCr & RList.Count & vbCr &  ↵
           QList.Count & vbCr & LList.Count & vbCr & CList.Count, "E, I, R, Q, L, C")
1520
1521            If IsBadRecords Then
1522                'find and display bad or corrupted Report entries
1523                Dim badList As List(Of String) = report.FindAll(AddressOf IsBadEntry)
1524                Dim badListStr As String = ""
1525
1526                For i As Integer = 1 To badList.Count
1527                    badListStr &= badList.Item(i - 1) & vbCrLf
1528                Next
1529
1530                Using sw As New IO.StreamWriter(unit.InstallPath & "Bad Records\error" & unit.UnitNum & _
1531                " - " & hostDateAvg.ToLongDateString & ".txt", True)
1532                    sw.WriteLine(hostDateAvg.ToString)
1533                    sw.WriteLine()
1534                    sw.Write(badListStr)
1535                    'sw.WriteLine()
1536                    sw.WriteLine("----------------------------------------------------------------")
1537                    sw.WriteLine()
1538                End Using
1539
1540                Notify("Warning:" & vbLf)
1541                Notify(vbLf)
1542                Notify("The following records from unit " & unit.UnitNum & " are invalid:" & vbLf)
1543                Notify(vbLf)
1544                Notify(badListStr & vbLf & vbLf)
1545                Notify("These records have been written to the file " & unit.InstallPath & _
1546                "Bad Records\error" & unit.UnitNum & " - " & _
1547                hostDateAvg.ToLongDateString & ".txt" & vbLf & vbLf)
1548                Notify("Trying download again..." & vbLf)
1549
1550                downloadAgain = True
1551            End If
1552
1553            If downloadAgain Then
1554                If unit.DLTries < 4 Then
1555                    'try to cancel the current download
1556                    Write("P999" & vbCr)
1557                    Thread.Sleep(1000)
1558                    Write("P999" & vbCr)
1559                    Thread.Sleep(1000)
1560                    comx.DiscardInBuffer()
```

```vb
1561                    'check to see if unit is still connected
1562                    Write("R1" & vbCr)
1563                    result = ReadLine(CChar(vbLf))
1564                    If result Like "*Larson*" Then
1565                        unit.DLTries += 1
1566                        Download(unit, False, worker, e)
1567                        Exit Sub
1568                    Else
1569                        unit.IsDownloadInProgress = False
1570                        unit.IsDownloadDone = False
1571                        Notify("Warning! Unit " & unit.UnitNum & " has timed out." & vbCr)
1572                        Notify("Download will resume after the other units have finished." & vbCr)
1573
1574                        'return unit to the LDUnitList,
1575                        '(it will be downloaded at a later point in time, after other units have been          ↵
         downloaded)
1576                        deleg2.Invoke(unit)
1577                        Exit Sub
1578                    End If
1579                Else
1580                    unit.IsDownloadInProgress = False
1581                    unit.IsDownloadDone = False
1582                    unit.DLTries = 0
1583                    Notify("Warning! Unit " & unit.UnitNum & " has a bad connection" & vbLf)
1584                    Notify("   and cannot transfer its data." & vbLf)
1585                    Notify("Download will resume after the other units have finished." & vbLf)
1586                    deleg2.Invoke(unit)
1587                    Exit Sub
1588                End If
1589            End If
1590
1591
1592            'Attempt to append corrected time to the E and I records
1593            'Note: unit time is returned in the format "Wed 28Jun2006 13:39:15" (with ends trimmed)
1594
1595            Dim RT22 As ReferenceTime = unit.GetReferenceTime() 'RT22 is the reference time stored on disk
1596            Dim HTcheck As New TimeSpan(hostDateAvg.Ticks - RT22.HostTimeZero.Ticks)
1597
1598            'MessageBox.Show(RT.UnitTimeZero & vbLf & RT.HostTimeZero, "Recorded Unit, Host Time")
1599
1600            If HTcheck.Days < 365 Then       'a previous useful reference time was found
1601
1602                Dim tDriftNum As Long = ((hostDateAvg.Ticks - unitDateAvg.Ticks) - _
1603                                (RT22.HostTimeZero.Ticks - RT22.UnitTimeZero.Ticks))
1604                Dim tDriftDenom As Long = (hostDateAvg.Ticks - RT22.HostTimeZero.Ticks)
1605                Dim timeDriftRatio As Double = (tDriftNum / tDriftDenom)
1606
1607                'MessageBox.Show(tDriftNum.ToString & vbLf & tDriftDenom.ToString & vbLf & timeDriftRatio.      ↵
         ToString, "Time Drift")
1608
1609                For i As Integer = 0 To EList.Count - 1
1610                    Dim subitem As String = EList.Item(i).Substring(7, 19)
1611                    Dim itemTime As Date = Date.Parse(subitem)
1612                    'MessageBox.Show(itemTime.ToString & vbLf & RT22.UnitTimeZero.ToString)
1613
1614                    If itemTime.Ticks >= RT22.UnitTimeZero.Ticks Then
1615                        'only perform time correction on data taken after the last 870 clock reset
1616                        Dim offsetZero As Long = RT22.HostTimeZero.Ticks - RT22.UnitTimeZero.Ticks
1617                        Dim Te_minus_To As Long = (itemTime.Ticks - RT22.UnitTimeZero.Ticks)
1618
1619                        Dim drift As New TimeSpan(offsetZero + _
1620                        CLng(Math.Round(Te_minus_To * timeDriftRatio, 0, MidpointRounding.AwayFromZero)))
1621
1622                        Dim correctedTime As Date = itemTime.Add(drift)
1623                        EList.Item(i) = EList.Item(i).TrimEnd(trimChars)
1624                        EList.Item(i) &= ", " & correctedTime.ToString("ddMMMyyyy HH:mm:ss.fff")
1625                    End If
1626                Next
1627
1628                For i As Integer = 0 To IList.Count - 1
1629                    Dim subitem As String = IList.Item(i).Substring(7, 19)
1630                    Dim itemTime As Date = Date.Parse(subitem)
1631                    'MessageBox.Show(itemTime.ToString & vbLf & RT22.UnitTimeZero.ToString)
1632
1633                    If itemTime.Ticks >= RT22.UnitTimeZero.Ticks Then
1634                        'only perform time correction on data taken after the last 870 clock reset
1635                        Dim offsetZero As Long = RT22.UnitTimeZero.Ticks - RT22.HostTimeZero.Ticks
1636                        Dim Te_minus_To As Long = itemTime.Ticks - RT22.UnitTimeZero.Ticks
1637
1638                        Dim drift As New TimeSpan(offsetZero + _
1639                        CLng(Math.Round(Te_minus_To * timeDriftRatio, 0, MidpointRounding.AwayFromZero)))
1640
1641                        Dim correctedTime As Date = itemTime.Add(drift)
1642                        IList.Item(i) = IList.Item(i).TrimEnd(trimChars)
1643                        IList.Item(i) &= ", " & correctedTime.ToString("ddMMMyyyy HH:mm:ss.fff")
1644                        'MessageBox.Show(itemTime.ToString & vbLf & correctedTime.ToString, "Uncorrected,       ↵
         Corrected")
1645                    End If
1646                Next
1647            End If
1648
1649
```

```vb
1650
1651            'Store report data somewhere
1652            '(if the download routine made it this far, the data is good)
1653
1654            unit.ErrorString = unitErrorString  'R98
1655            unit.EList = EList
1656            unit.IList = IList
1657            unit.NumExceedances = EList.Count
1658            unit.NumIntervals = IList.Count
1659            unit.BattVoltage = battVoltage
1660            unit.LastDL = hostDateAvg
1661            unit.NumStartStops = numStartStops
1662            unit.NumCalibrations = numCalibrations
1663            unit.CalLevel = calLevel
1664            If unit.IncludeR Then
1665                unit.RList = RList
1666            End If
1667            If unit.IncludeQ Then
1668                unit.QList = QList
1669            End If
1670            If unit.IncludeLC Then
1671                unit.LList = LList
1672                unit.CList = CList
1673            End If
1674
1675            unit.SerializeReports()
1676
1677            Dim sqlerror As String = unit.SaveToDB(RTcurrent)
1678            If Not sqlerror Like "" Then
1679                Notify(sqlerror & vbLf)
1680            End If
1681
1682
1683
1684            'Reset unit time
1685            If unit.ResetTimeYN Then
1686
1687                Write("M4" & vbCr)
1688                result = ReadLine(CChar(vbLf))
1689                result = result.Trim()
1690                If Not result Like "" Then
1691                    Write("M4" & vbCr)
1692                    result = ReadLine(CChar(vbLf))
1693                    result = result.Trim()
1694                    If Not result Like "" Then
1695                        Write("M4" & vbCr)
1696                        result = ReadLine(CChar(vbLf))
1697                    End If
1698                End If
1699
1700                Dim newUnitTime As Date = Date.Now
1701                Write("S6," & newUnitTime.ToString("HH:mm:ss") & vbCr)
1702                result = ReadLine(CChar(vbLf))
1703                result = result.Trim()
1704                If result Like "*Stop*" Then
1705                    Write("M4" & vbCr)
1706                    result = ReadLine(CChar(vbLf))
1707                    result = result.Trim()
1708                    If Not result Like "" Then
1709                        Write("M4" & vbCr)
1710                        result = ReadLine(CChar(vbLf))
1711                        result = result.Trim()
1712                        If Not result Like "" Then
1713                            Write("M4" & vbCr)
1714                            result = ReadLine(CChar(vbLf))
1715                        End If
1716                    End If
1717                    newUnitTime = Date.Now
1718                    Write("S6," & newUnitTime.ToString("HH:mm:ss") & vbCr)
1719                    result = ReadLine(CChar(vbLf))
1720                End If
1721                If Not result Like "" Then
1722                    newUnitTime = Date.Now
1723                    Write("S6," & newUnitTime.ToString("HH:mm:ss") & vbCr)
1724                    result = ReadLine(CChar(vbLf))
1725                End If
1726                Write("M3" & vbCr)
1727                result = ReadLine(CChar(vbLf))
1728                result = result.Trim()
1729                If Not result Like "" Then
1730                    Write("M3" & vbCr)
1731                    result = ReadLine(CChar(vbLf))
1732                    result = result.Trim()
1733                    If Not result Like "" Then
1734                        Write("M3" & vbCr)
1735                        result = ReadLine(CChar(vbLf))
1736                    End If
1737                End If
1738
1739                'take a new time difference, store it to disk to use to compute the time drift for future data
1740                'Get unit and host times just after the reset -
1741                'this will be the reference "Time Zero" for the time drift calculations
```

```vb
1742
1743                Dim RTzero As ReferenceTime = GetUnitTime(worker, e)
1744
1745                'Record the unit & host times
1746                unit.SerializeReferenceTime(RTzero)
1747                unit.ResetTimeYN = False
1748
1749            End If
1750
1751
1752            'Reset unit data
1753            If unit.ResetDataYN AndAlso sqlerror = "" Then
1754                Write("M4" & vbCr)
1755                result = ReadLine(CChar(vbLf))
1756                result = result.Trim()
1757                If Not result Like "" Then
1758                    Write("M4" & vbCr)
1759                    result = ReadLine(CChar(vbLf))
1760                    result = result.Trim()
1761                    If Not result Like "" Then
1762                        Write("M4" & vbCr)
1763                        result = ReadLine(CChar(vbLf))
1764                    End If
1765                End If
1766                Write("S1,1" & vbCr)
1767                result = ReadLine(CChar(vbLf))
1768                result = result.Trim()
1769                If result Like "*Stop*" Then
1770                    Write("M4" & vbCr)
1771                    result = ReadLine(CChar(vbLf))
1772                    result = result.Trim()
1773                    If Not result Like "" Then
1774                        Write("M4" & vbCr)
1775                        result = ReadLine(CChar(vbLf))
1776                        result = result.Trim()
1777                        If Not result Like "" Then
1778                            Write("M4" & vbCr)
1779                            result = ReadLine(CChar(vbLf))
1780                        End If
1781                    End If
1782                    Write("S1,1" & vbCr)
1783                    result = ReadLine(CChar(vbLf))
1784                End If
1785                Write("S1,1" & vbCr)
1786                result = ReadLine(CChar(vbLf))
1787
1788
1789                'change exceedance threshold if necessary
1790                'Dim desiredExcdThresh As Integer = 0
1791                'Dim now As Date = Date.Now
1792                'Dim nowDay As DayOfWeek = now.DayOfWeek
1793                'Dim nowTime As TimeSpan = now.TimeOfDay
1794                'Dim nowtimesec As Long = CLng(nowTime.TotalSeconds)  'truncate the fractional seconds
1795                'nowTime = New TimeSpan(nowtimesec * 10000000)
1796
1797                'Select Case nowDay
1798                '    Case DayOfWeek.Sunday
1799                '        desiredExcdThresh = unit.ExcdNight
1800                '    Case DayOfWeek.Saturday
1801                '        If Form1.appset.IsSatAWorkday Then
1802                '            If Form1.appset.WorkdayStart <= Form1.appset.WorkdayEnd Then
1803                '                If nowTime > Form1.appset.WorkdayStart AndAlso nowTime < Form1.appset.↵
       WorkdayEnd Then
1804                '                    desiredExcdThresh = unit.ExcdDay
1805                '                Else
1806                '                    desiredExcdThresh = unit.ExcdNight
1807                '                End If
1808                '            Else
1809                '                If nowTime > Form1.appset.WorkdayEnd AndAlso nowTime < Form1.appset.↵
       WorkdayStart Then
1810                '                    desiredExcdThresh = unit.ExcdNight
1811                '                Else
1812                '                    desiredExcdThresh = unit.ExcdDay
1813                '                End If
1814                '            End If
1815                '        Else
1816                '            desiredExcdThresh = unit.ExcdNight
1817                '        End If
1818
1819                '        'otherwise
1820                '    Case Else
1821                '        If Form1.appset.WorkdayStart <= Form1.appset.WorkdayEnd Then
1822                '            If nowTime > Form1.appset.WorkdayStart AndAlso nowTime < Form1.appset.WorkdayEnd↵
       Then
1823                '                desiredExcdThresh = unit.ExcdDay
1824                '            Else
1825                '                desiredExcdThresh = unit.ExcdNight
1826                '            End If
1827                '        Else 'ex. day starts at 6am and ends at 1am
1828                '            If nowTime > Form1.appset.WorkdayEnd AndAlso nowTime < Form1.appset.WorkdayStart↵
       Then
1829                '                desiredExcdThresh = unit.ExcdNight
```

```vb
1830                '            Else
1831                '                desiredExcdThresh = unit.ExcdDay
1832                '            End If
1833                '        End If
1834                'End Select
1835
1836                Dim desiredExcdThresh As Integer = 0
1837                If Not Form1.appset.StopAtNight Then
1838                    'replace above with  Not Form1.appset.StopAtNight  later
1839                    'Aberdeen only is test - soon all units won't run at night
1840                    If Form1.IsDaytime Then
1841                        desiredExcdThresh = unit.ExcdDay
1842                    Else
1843                        desiredExcdThresh = unit.ExcdNight
1844                    End If
1845                Else
1846                    desiredExcdThresh = unit.ExcdDay 'since unit won't run at night
1847                End If
1848
1849                If desiredExcdThresh <> CInt(excdThreshTemp) AndAlso desiredExcdThresh > 29 Then
1850                    If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
1851                        Write("S64," & desiredExcdThresh.ToString & vbCr)
1852                        result = ReadLine(CChar(vbLf))
1853                        Write("S64," & desiredExcdThresh.ToString & vbCr)
1854                        result = ReadLine(CChar(vbLf))
1855                    Else
1856                        Write("S64," & desiredExcdThresh.ToString & vbCr)
1857                        result = ReadLine(CChar(vbLf))
1858                        Write("S64," & desiredExcdThresh.ToString & vbCr)
1859                        result = ReadLine(CChar(vbLf))
1860                    End If
1861                    Write("Q64" & vbCr)
1862                    result = ReadLine(CChar(vbLf))
1863                    result = result.Trim(trimChars)
1864                    If result Like "*ARNING*" Then
1865                        Write("Q64" & vbCr)
1866                        result = ReadLine(CChar(vbLf))
1867                        result = result.Trim(trimChars)
1868                    End If
1869                    If result Like "*No response.*" Then  'ReadLine timed out
1870                        Dim tempd, tempd2 As Double
1871
1872                        comx.DiscardInBuffer()
1873                        Write("Q64" & vbCr)
1874                        result = ReadLine(CChar(vbLf))
1875                        result = result.Trim(trimChars)
1876                        result2 = ReadLine(CChar(vbLf), 1)
1877                        result2 = result2.Trim(trimChars)
1878                        parseRes = Double.TryParse(result, tempd)
1879                        parseRes2 = Double.TryParse(result2, tempd2)
1880                        If parseRes2 Then        'use second read entry
1881                            excdThreshTemp = tempd2
1882                        ElseIf parseRes Then     'use first read entry
1883                            excdThreshTemp = tempd
1884                        End If
1885                    Else
1886                        excdThreshTemp = Double.Parse(result)
1887                    End If
1888                    If desiredExcdThresh <> CInt(excdThreshTemp) AndAlso desiredExcdThresh > 29 Then
1889                        If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
1890                            Write("S64," & desiredExcdThresh.ToString & vbCr)
1891                            result = ReadLine(CChar(vbLf))
1892                            Write("S64," & desiredExcdThresh.ToString & vbCr)
1893                            result = ReadLine(CChar(vbLf))
1894                        Else
1895                            Write("S64," & desiredExcdThresh.ToString & vbCr)
1896                            result = ReadLine(CChar(vbLf))
1897                            Write("S64," & desiredExcdThresh.ToString & vbCr)
1898                            result = ReadLine(CChar(vbLf))
1899                        End If
1900                    End If
1901                End If
1902
1903                Write("M3" & vbCr)
1904                result = ReadLine(CChar(vbLf))
1905                result = result.Trim()
1906                If Not result Like "" Then
1907                    Write("M3" & vbCr)
1908                    result = ReadLine(CChar(vbLf))
1909                    result = result.Trim()
1910                    If Not result Like "" Then
1911                        Write("M3" & vbCr)
1912                        result = ReadLine(CChar(vbLf))
1913                    End If
1914                End If
1915            End If
1916
1917            'store exceedance threshold
1918            unit.ExcdThreshold = CInt(excdThreshTemp)
1919
1920
1921
```

```vb
1922
1923            'Tell unit to dial out on both exceedance and alarm (such as low battery)
1924            If unit.AllowCallIns Then
1925                Write("Q156" & vbCr)
1926                result = ReadLine(CChar(vbLf))
1927                result = result.Trim(trimChars)
1928                If Not String.Compare(result, "T" & modemPhoneNum) = 0 AndAlso Not unit.UnitNum = "L99" Then
1929                    'test - don't reset phone # on unit 99
1930                    Write("S156,T" & modemPhoneNum & vbCr)
1931                    result = ReadLine(CChar(vbLf))
1932                    Write("S156,T" & modemPhoneNum & vbCr)
1933                    result = ReadLine(CChar(vbLf))
1934                End If
1935                Write("S155,3" & vbCr)
1936                result = ReadLine(CChar(vbLf))
1937                Write("S155,3" & vbCr)
1938                result = ReadLine(CChar(vbLf))
1939            Else    'uncomment for in-house unit only
1940                Write("S155,0" & vbCr)
1941                result = ReadLine(CChar(vbLf))
1942                Write("S155,0" & vbCr)
1943                result = ReadLine(CChar(vbLf))
1944            End If
1945
1946
1947
1948            'Set unit's run and stop timer (if necessary)
1949            If unit.TimerChanged Then
1950                Write("S24," & unit.TimerRun1 & vbCr)
1951                result = ReadLine(CChar(vbLf))
1952                Write("S24," & unit.TimerRun1 & vbCr)
1953                result = ReadLine(CChar(vbLf))
1954                Write("S25," & unit.TimerStop1 & vbCr)
1955                result = ReadLine(CChar(vbLf))
1956                Write("S25," & unit.TimerStop1 & vbCr)
1957                result = ReadLine(CChar(vbLf))
1958                Write("Q24" & vbCr)
1959                result = ReadLine(CChar(vbLf))
1960                result = result.Trim(trimChars)
1961                If Not result = unit.TimerRun1 Then
1962                    Write("S24," & unit.TimerRun1 & vbCr)
1963                    result = ReadLine(CChar(vbLf))
1964                    Write("S24," & unit.TimerRun1 & vbCr)
1965                    result = ReadLine(CChar(vbLf))
1966                End If
1967                Write("Q25" & vbCr)
1968                result = ReadLine(CChar(vbLf))
1969                result = result.Trim(trimChars)
1970                If Not result = unit.TimerStop1 Then
1971                    Write("S25," & unit.TimerStop1 & vbCr)
1972                    result = ReadLine(CChar(vbLf))
1973                    Write("S25," & unit.TimerStop1 & vbCr)
1974                    result = ReadLine(CChar(vbLf))
1975                End If
1976                unit.TimerChanged = False
1977            End If
1978
1979
1980
1981            'Send custom parameters
1982            If unit.SendCustP Then
1983                Notify("*** Custom parameters ***" & vbLf)
1984                Dim seps() As Char = {CChar(vbLf), CChar(vbCr)}
1985                Dim cmds() As String = unit.CustParams.Split(seps, StringSplitOptions.RemoveEmptyEntries)
1986                For Each cmd As String In cmds
1987                    cmd.Trim()
1988                    Write(cmd & vbCr)
1989                    result = ReadLine(CChar(vbLf))
1990                    'result = result.Trim(trimChars)
1991                    If Not (result Like "*" & vbLf & "*" OrElse result Like "*" & vbCr & "*") Then
1992                        Write(cmd & vbCr)
1993                        result = ReadLine(CChar(vbLf))
1994                    End If
1995                Next
1996                Notify("*** End of custom parameters ***" & vbLf)
1997                unit.SendCustP = False
1998            End If
1999
2000
2001
2002            'check to see if unit is running
2003            'if nighttime, stop unit (M4); otherwise, run unit (M3)
2004
2005            Write("R3" & vbCr)
2006            result = ReadLine(CChar(vbLf))
2007            If Form1.appset.StopAtNight AndAlso Not Form1.IsDaytime Then   'stop unit
2008                If Not result Like "*Stop*" Then
2009                    Write("M4" & vbCr)
2010                    result = ReadLine(CChar(vbLf))
2011                    result = result.Trim()
2012                    If Not result Like "" Then
2013                        Write("M4" & vbCr)
```

```vb
2014                           result = ReadLine(CChar(vbLf))
2015                           result = result.Trim()
2016                           If Not result Like "" Then
2017                               Write("M4" & vbCr)
2018                               result = ReadLine(CChar(vbLf))
2019                           End If
2020                       End If
2021                       Write("R3" & vbCr)
2022                       result = ReadLine(CChar(vbLf))
2023                       If Not result Like "*Stop*" Then
2024                           Write("M4" & vbCr)
2025                           result = ReadLine(CChar(vbLf))
2026                           result = result.Trim()
2027                           If Not result Like "" Then
2028                               Write("M4" & vbCr)
2029                               result = ReadLine(CChar(vbLf))
2030                               result = result.Trim()
2031                               If Not result Like "" Then
2032                                   Write("M4" & vbCr)
2033                                   result = ReadLine(CChar(vbLf))
2034                               End If
2035                           End If
2036                       End If
2037                   End If
2038           Else   'run unit
2039               If Not result Like "*Run*" Then
2040                   Write("M3" & vbCr)
2041                   result = ReadLine(CChar(vbLf))
2042                   result = result.Trim()
2043                   If Not result Like "" Then
2044                       Write("M3" & vbCr)
2045                       result = ReadLine(CChar(vbLf))
2046                       result = result.Trim()
2047                       If Not result Like "" Then
2048                           Write("M3" & vbCr)
2049                           result = ReadLine(CChar(vbLf))
2050                       End If
2051                   End If
2052                   Write("R3" & vbCr)
2053                   result = ReadLine(CChar(vbLf))
2054                   If Not result Like "*Run*" Then
2055                       Write("M3" & vbCr)
2056                       result = ReadLine(CChar(vbLf))
2057                       result = result.Trim()
2058                       If Not result Like "" Then
2059                           Write("M3" & vbCr)
2060                           result = ReadLine(CChar(vbLf))
2061                           result = result.Trim()
2062                           If Not result Like "" Then
2063                               Write("M3" & vbCr)
2064                               result = ReadLine(CChar(vbLf))
2065                           End If
2066                       End If
2067                   End If
2068               End If
2069           End If
2070
2071
2072
2073           'Store report data somewhere
2074           '(if the download routine made it this far, the data is good)
2075
2076           unit.ErrorString = unitErrorString  'R98
2077           unit.EList = EList
2078           unit.IList = IList
2079           unit.NumExceedances = EList.Count
2080           unit.NumIntervals = IList.Count
2081           unit.BattVoltage = battVoltage
2082           unit.LastDL = hostDateAvg
2083           unit.NumStartStops = numStartStops
2084           unit.NumCalibrations = numCalibrations
2085           'unit.ExcdThreshold = CInt(excdThreshTemp)
2086           unit.CalLevel = calLevel
2087
2088
2089
2090           'Download complete
2091           Write("M12" & vbCr)
2092           Notify("*** Download of unit " & unit.UnitNum & " is complete ***" & vbCr)
2093           unit.IsDownloadInProgress = False
2094           unit.IsDownloadDone = True
2095           unit.DLTries = 0
2096           unit.DialTries = 0
2097           If Not sqlerror Like "" Then
2098               unit.IsDownloadDone = False
2099           End If
2100           If Not (unit.UnitNum = "null" OrElse unit.UnitNum = "temp") Then
2101               'If unit.IsDownloadDone = True AndAlso Me.modemMode = Mode.Dial Then
2102               If Form1.appset.DialInProg Then 'remove unit from the status bar lists - even if it dialed in
2103                   Dim SBarEA As New StatusBarEventArgs(unit.UnitNum, StatusBarEventArgs.SBAction.Remove)
2104                   'RaiseEvent UpdateStatusBar(Me, SBarEA)
2105                   RaiseEvent SBUnitCounter(Me, SBarEA)
```

```vb
2106                    RaiseEvent UpdateMessages(Me, SBarEA)
2107                End If
2108                'If Me.modemMode = Mode.Listen Then   'make the unit available to the list
2109                '    unit.IsDownloadDone = False
2110                '    unit.IsDownloadInProgress = False
2111                'End If
2112                deleg2.Invoke(unit)
2113            End If
2114        End Sub
2115
2116        Public Sub ReconfigureUnit(ByVal unit As LDUnit, ByVal askPass As Boolean, _
2117        ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs)
2118
2119            'Reconfigures the report settings on the unit
2120
2121            Dim result As New List(Of String)
2122            Dim unitnumstr As String = unit.UnitNum.Substring(1)     'remove the leading "L"
2123            Dim unitNum As Integer = Integer.Parse(unitnumstr)
2124            Dim isConfigGood As Boolean = True
2125
2126            If askPass Then
2127                unit = getPass(unit, worker, e)
2128            End If
2129
2130            Notify("Reconfiguring report settings..." & vbLf)
2131            If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
2132                'set up Aberdeen units
2133                Write("M4" & vbCr)
2134                result.Add(ReadLine(CChar(vbLf)))
2135                Write("S1,1" & vbCr)
2136                result.Add(ReadLine(CChar(vbLf)))
2137                Write("S37,1" & vbCr)
2138                result.Add(ReadLine(CChar(vbLf)))
2139                Write("S40,2" & vbCr)
2140                result.Add(ReadLine(CChar(vbLf)))
2141                Write("S44,0" & vbCr)
2142                result.Add(ReadLine(CChar(vbLf)))
2143                Write("S61,200" & vbCr)
2144                result.Add(ReadLine(CChar(vbLf)))
2145                Write("S62,1" & vbCr)
2146                result.Add(ReadLine(CChar(vbLf)))
2147                Write("S69,0" & vbCr)
2148                result.Add(ReadLine(CChar(vbLf)))
2149                Write("S71,1" & vbCr)
2150                result.Add(ReadLine(CChar(vbLf)))
2151                Write("S78,1" & vbCr)
2152                result.Add(ReadLine(CChar(vbLf)))
2153                Write("S80,0" & vbCr)
2154                result.Add(ReadLine(CChar(vbLf)))
2155                Write("S88,0" & vbCr)
2156                result.Add(ReadLine(CChar(vbLf)))
2157                Write("S89,1" & vbCr)
2158                result.Add(ReadLine(CChar(vbLf)))
2159                Write("S90,1" & vbCr)
2160                result.Add(ReadLine(CChar(vbLf)))
2161                Write("S91,1" & vbCr)
2162                result.Add(ReadLine(CChar(vbLf)))
2163                Write("S92,0" & vbCr)
2164                result.Add(ReadLine(CChar(vbLf)))
2165                Write("S93,40" & vbCr)
2166                result.Add(ReadLine(CChar(vbLf)))
2167                Write("S94,90" & vbCr)
2168                result.Add(ReadLine(CChar(vbLf)))
2169                Write("S95,4" & vbCr)
2170                result.Add(ReadLine(CChar(vbLf)))
2171                Write("S96,0" & vbCr)
2172                result.Add(ReadLine(CChar(vbLf)))
2173                Write("S97,70" & vbCr)
2174                result.Add(ReadLine(CChar(vbLf)))
2175                Write("S98,120" & vbCr)
2176                result.Add(ReadLine(CChar(vbLf)))
2177                Write("S99,4" & vbCr)
2178                result.Add(ReadLine(CChar(vbLf)))
2179                Write("S100,0" & vbCr)
2180                result.Add(ReadLine(CChar(vbLf)))
2181                Write("S101,110" & vbCr)
2182                result.Add(ReadLine(CChar(vbLf)))
2183                Write("S102,160" & vbCr)
2184                result.Add(ReadLine(CChar(vbLf)))
2185                Write("S103,4" & vbCr)
2186                result.Add(ReadLine(CChar(vbLf)))
2187                Write("S104,1" & vbCr)
2188                result.Add(ReadLine(CChar(vbLf)))
2189                Write("S105,0" & vbCr)
2190                result.Add(ReadLine(CChar(vbLf)))
2191                Write("S106,65535" & vbCr)
2192                result.Add(ReadLine(CChar(vbLf)))
2193                Write("S107,1" & vbCr)
2194                result.Add(ReadLine(CChar(vbLf)))
2195                Write("S108,0" & vbCr)
2196                result.Add(ReadLine(CChar(vbLf)))
2197                Write("S109,65535" & vbCr)
```

```vb
2198                result.Add(ReadLine(CChar(vbLf)))
2199                Write("S110,1" & vbCr)
2200                result.Add(ReadLine(CChar(vbLf)))
2201                Write("S111,0" & vbCr)
2202                result.Add(ReadLine(CChar(vbLf)))
2203                Write("S113,1" & vbCr)
2204                result.Add(ReadLine(CChar(vbLf)))
2205                Write("S118,0" & vbCr)
2206                result.Add(ReadLine(CChar(vbLf)))
2207                Write("S119,0" & vbCr)
2208                result.Add(ReadLine(CChar(vbLf)))
2209                Write("S120,0" & vbCr)
2210                result.Add(ReadLine(CChar(vbLf)))
2211                Write("S121,Wind" & vbCr)
2212                result.Add(ReadLine(CChar(vbLf)))
2213                Write("S122,0" & vbCr)
2214                result.Add(ReadLine(CChar(vbLf)))
2215                Write("S123,0" & vbCr)
2216                result.Add(ReadLine(CChar(vbLf)))
2217                Write("S124,Knot" & vbCr)
2218                result.Add(ReadLine(CChar(vbLf)))
2219                Write("S125,0" & vbCr)
2220                result.Add(ReadLine(CChar(vbLf)))
2221                Write("S140,RH" & vbCr)
2222                result.Add(ReadLine(CChar(vbLf)))
2223                Write("S149,008.1" & vbCr)
2224                result.Add(ReadLine(CChar(vbLf)))
2225                Write("S150,1" & vbCr)
2226                result.Add(ReadLine(CChar(vbLf)))
2227                Write("S151,0" & vbCr)
2228                result.Add(ReadLine(CChar(vbLf)))
2229                Write("S152,0" & vbCr)
2230                result.Add(ReadLine(CChar(vbLf)))
2231                Write("S153,1" & vbCr)
2232                result.Add(ReadLine(CChar(vbLf)))
2233                Write("S155,3" & vbCr)
2234                result.Add(ReadLine(CChar(vbLf)))
2235                Write("S156,14102721390" & vbCr)
2236                result.Add(ReadLine(CChar(vbLf)))
2237                Write("S157," & unitNum & vbCr)
2238                result.Add(ReadLine(CChar(vbLf)))
2239                'Write("s158,X4 E0 Q0 V0 T M1 S0=5 S10=100 &D3" & vbCr)
2240                'result.add(readline(cchar(vblf)))
2241                Write("S159,1" & vbCr)
2242                result.Add(ReadLine(CChar(vbLf)))
2243                Write("S160,10.8" & vbCr)
2244                result.Add(ReadLine(CChar(vbLf)))
2245                Write("S161,00:03" & vbCr)
2246                result.Add(ReadLine(CChar(vbLf)))
2247                Write("S167,0406002701890A3D0666" & vbCr)
2248                result.Add(ReadLine(CChar(vbLf)))
2249                Write("S168,0" & vbCr)
2250                result.Add(ReadLine(CChar(vbLf)))
2251                Write("S169,0" & vbCr)
2252                result.Add(ReadLine(CChar(vbLf)))
2253                Write("S170,0" & vbCr)
2254                result.Add(ReadLine(CChar(vbLf)))
2255                Write("S171,0" & vbCr)
2256                result.Add(ReadLine(CChar(vbLf)))
2257                Write("S172,0" & vbCr)
2258                result.Add(ReadLine(CChar(vbLf)))
2259                Write("S173,0" & vbCr)
2260                result.Add(ReadLine(CChar(vbLf)))
2261                Write("S174,1" & vbCr)
2262                result.Add(ReadLine(CChar(vbLf)))
2263                Write("S192,2" & vbCr)
2264                result.Add(ReadLine(CChar(vbLf)))
2265                Write("S193,0" & vbCr)
2266                result.Add(ReadLine(CChar(vbLf)))
2267                Write("S194,65535" & vbCr)
2268                result.Add(ReadLine(CChar(vbLf)))
2269                Write("S195,2" & vbCr)
2270                result.Add(ReadLine(CChar(vbLf)))
2271                Write("S196,0" & vbCr)
2272                result.Add(ReadLine(CChar(vbLf)))
2273                Write("S197,65535" & vbCr)
2274                result.Add(ReadLine(CChar(vbLf)))
2275                'Write("S201,2" & vbCr)
2276                'result.Add(ReadLine(CChar(vbLf)))
2277                'Write("S202,0" & vbCr)
2278                'result.Add(ReadLine(CChar(vbLf)))
2279                'Write("S207,8" & vbCr)
2280                'result.Add(ReadLine(CChar(vbLf)))
2281                'Write("S210,49" & vbCr)
2282                'result.Add(ReadLine(CChar(vbLf)))
2283                Write("M3" & vbCr)
2284                result.Add(ReadLine(CChar(vbLf)))
2285            Else
2286                'set up CERL units
2287                Write("M4" & vbCr)
2288                result.Add(ReadLine(CChar(vbLf)))
2289                Write("S1,1" & vbCr)
```

```vb
2290                result.Add(ReadLine(CChar(vbLf)))
2291                Write("S12,5" & vbCr)
2292                result.Add(ReadLine(CChar(vbLf)))
2293                Write("S16,0" & vbCr)
2294                result.Add(ReadLine(CChar(vbLf)))
2295                Write("S38,06:00" & vbCr)
2296                result.Add(ReadLine(CChar(vbLf)))
2297
2298                Write("S61,200" & vbCr)
2299                result.Add(ReadLine(CChar(vbLf)))
2300                Write("S62,200" & vbCr)
2301                result.Add(ReadLine(CChar(vbLf)))
2302                Write("S63,100" & vbCr)
2303                result.Add(ReadLine(CChar(vbLf)))
2304                Write("S64,200" & vbCr)
2305                result.Add(ReadLine(CChar(vbLf)))
2306                Write("S69,0" & vbCr)
2307                result.Add(ReadLine(CChar(vbLf)))
2308                Write("S71,1" & vbCr)
2309                result.Add(ReadLine(CChar(vbLf)))
2310                Write("S78,1" & vbCr)
2311                result.Add(ReadLine(CChar(vbLf)))
2312                Write("S80,0" & vbCr)
2313                result.Add(ReadLine(CChar(vbLf)))
2314                Write("S88,0" & vbCr)
2315                result.Add(ReadLine(CChar(vbLf)))
2316                Write("S89,1" & vbCr)
2317                result.Add(ReadLine(CChar(vbLf)))
2318                Write("S90,1" & vbCr)
2319                result.Add(ReadLine(CChar(vbLf)))
2320                Write("S91,1" & vbCr)
2321                result.Add(ReadLine(CChar(vbLf)))
2322                Write("S92,0" & vbCr)
2323                result.Add(ReadLine(CChar(vbLf)))
2324                Write("S96,0" & vbCr)
2325                result.Add(ReadLine(CChar(vbLf)))
2326                Write("S99,4" & vbCr)
2327                result.Add(ReadLine(CChar(vbLf)))
2328                Write("S100,0" & vbCr)
2329                result.Add(ReadLine(CChar(vbLf)))
2330                Write("S104,1" & vbCr)
2331                result.Add(ReadLine(CChar(vbLf)))
2332                Write("S105,0" & vbCr)
2333                result.Add(ReadLine(CChar(vbLf)))
2334                Write("S106,65535" & vbCr)
2335                result.Add(ReadLine(CChar(vbLf)))
2336                Write("S107,1" & vbCr)
2337                result.Add(ReadLine(CChar(vbLf)))
2338                Write("S108,0" & vbCr)
2339                result.Add(ReadLine(CChar(vbLf)))
2340                Write("S109,65535" & vbCr)
2341                result.Add(ReadLine(CChar(vbLf)))
2342                Write("S110,1" & vbCr)
2343                result.Add(ReadLine(CChar(vbLf)))
2344                Write("S111,0" & vbCr)
2345                result.Add(ReadLine(CChar(vbLf)))
2346                Write("S113,1" & vbCr)
2347                result.Add(ReadLine(CChar(vbLf)))
2348                Write("S118,0" & vbCr)
2349                result.Add(ReadLine(CChar(vbLf)))
2350                Write("S119,0" & vbCr)
2351                result.Add(ReadLine(CChar(vbLf)))
2352                Write("S120,0" & vbCr)
2353                result.Add(ReadLine(CChar(vbLf)))
2354                Write("S150,1" & vbCr)
2355                result.Add(ReadLine(CChar(vbLf)))
2356                Write("S151,0" & vbCr)
2357                result.Add(ReadLine(CChar(vbLf)))
2358                Write("S152,0" & vbCr)
2359                result.Add(ReadLine(CChar(vbLf)))
2360                Write("S153,1" & vbCr)
2361                result.Add(ReadLine(CChar(vbLf)))
2362                Write("S155,0" & vbCr)
2363                result.Add(ReadLine(CChar(vbLf)))
2364                Write("S160,09.5" & vbCr)
2365                result.Add(ReadLine(CChar(vbLf)))
2366                Write("S168,0" & vbCr)
2367                result.Add(ReadLine(CChar(vbLf)))
2368                Write("S169,0" & vbCr)
2369                result.Add(ReadLine(CChar(vbLf)))
2370                Write("S170,0" & vbCr)
2371                result.Add(ReadLine(CChar(vbLf)))
2372                Write("S171,0" & vbCr)
2373                result.Add(ReadLine(CChar(vbLf)))
2374                Write("S172,0" & vbCr)
2375                result.Add(ReadLine(CChar(vbLf)))
2376                Write("S173,0" & vbCr)
2377                result.Add(ReadLine(CChar(vbLf)))
2378                Write("S174,1" & vbCr)
2379                result.Add(ReadLine(CChar(vbLf)))
2380                Write("S192,2" & vbCr)
2381                result.Add(ReadLine(CChar(vbLf)))
```

```vb
2382                    Write("S193,0" & vbCr)
2383                    result.Add(ReadLine(CChar(vbLf)))
2384                    Write("S194,65535" & vbCr)
2385                    result.Add(ReadLine(CChar(vbLf)))
2386                    Write("S195,2" & vbCr)
2387                    result.Add(ReadLine(CChar(vbLf)))
2388                    Write("S196,0" & vbCr)
2389                    result.Add(ReadLine(CChar(vbLf)))
2390                    Write("S197,65535" & vbCr)
2391                    result.Add(ReadLine(CChar(vbLf)))
2392                    'Write("S201,2" & vbCr)
2393                    'result.Add(ReadLine(CChar(vbLf)))
2394                    'Write("S202,0" & vbCr)
2395                    'result.Add(ReadLine(CChar(vbLf)))
2396                    'Write("S207,008.00" & vbCr)
2397                    'result.Add(ReadLine(CChar(vbLf)))
2398                    'Write("S210,49" & vbCr)
2399                    'result.Add(ReadLine(CChar(vbLf)))
2400                    Write("M3" & vbCr)
2401                    result.Add(ReadLine(CChar(vbLf)))
2402                End If
2403
2404                If worker.CancellationPending Then
2405                    Write("M12" & vbCr)
2406                    result.Add(ReadLine(CChar(vbLf)))
2407                    Throw New CancelException
2408                End If
2409
2410                'Write("" & vbCr)
2411                'result.Add(ReadLine(CChar(vbLf)))
2412
2413                For Each item As String In result
2414                    item.Trim()
2415                    If item.Length > 4 Then      'an error occurred
2416                        isConfigGood = False
2417                    End If
2418                Next
2419
2420                If isConfigGood = False Then
2421                    ReconfigureUnit(unit, False, worker, e)    'repeat the configuration
2422                End If
2423
2424                unit.Initialize = False
2425            End Sub
2426
2427 #Region "Helper Methods"
2428
2429        Public Function getPass(ByVal unit As LDUnit, _
2430        ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs) As LDUnit
2431
2432            'unified way to get unit password
2433            Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
2434
2435            Dim result As String = ""
2436            Dim result2 As String = ""
2437            Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab)}
2438            Dim checkResponse As Boolean = False
2439
2440
2441            For i As Integer = 1 To 10
2442                result = ReadLine(":"c, 2)
2443                result = result.Trim(trimChars)
2444                'MessageBox.Show(result, "Read result")
2445                If result Like "*870?????*" Then
2446                    Dim pos As Integer = result.IndexOf("870")
2447                    Dim pass As String = result.Substring(pos, 8)
2448                    unit.UnitSerial = pass
2449                    Write(pass & ":" & unit.LockCode & vbCr)
2450                    result = ReadLine(CChar(vbLf), 2)
2451                    If result Like "*Ready*" Then
2452                        Exit For
2453                    End If
2454                ElseIf result Like "*ARNING*" Then      'a CERL unit
2455                    checkResponse = True
2456                    Exit For
2457                ElseIf result Like "*E*" OrElse result Like "*I*" Then
2458                    Write("P999" & vbCr)
2459                    'checkResponse = True
2460                    Exit For
2461                ElseIf i >= 10 Then  'see if unit is responsive
2462                    checkResponse = True
2463                End If
2464            Next
2465
2466            If checkResponse Then
2467                comx.DiscardInBuffer()
2468                Write("R1" & vbCr)
2469                result = ReadLine(CChar(vbLf))
2470                result = result.Trim(trimChars)
2471                Write("R1" & vbCr)
2472                result = ReadLine(CChar(vbLf))
2473                result = result.Trim(trimChars)
```

```vb
2474                 If Not result Like "*No response.*" Then   'unit is responsive - get unit serial
2475                     'result Like "*Larson*" OrElse result Like "*ARNING*" Then   'get unit serial #
2476                     comx.DiscardInBuffer()
2477                     Write("R89" & vbCr)
2478                     result = ReadLine(CChar(vbLf))
2479                     result = result.Trim(trimChars)
2480                     If result Like "*No response.*" Then  'ReadLine timed out
2481                         comx.DiscardInBuffer()
2482                         Write("R89" & vbCr)
2483                         result = ReadLine(CChar(vbLf))
2484                         result = result.Trim(trimChars)
2485                         result2 = ReadLine(CChar(vbLf), 1)
2486                         result2 = result2.Trim(trimChars)
2487                         If Not result Like "" OrElse Not result Like "*ARNING*" Then        'use first read entry
2488                             unit.UnitSerial = "870" & result.Trim(trimChars)
2489                         ElseIf Not result2 Like "" Then    'use second read entry
2490                             unit.UnitSerial = "870" & result2.Trim(trimChars)
2491                         Else
2492                             Throw New FormatException("Can't read from unit - will try again later")
2493                         End If
2494                     Else
2495                         unit.UnitSerial = "870" & result.Trim(trimChars)
2496                     End If
2497                 Else    'unit is not responsive - disconnect
2498                     unit.IsDownloadInProgress = False
2499                     Throw New Exception("Could not log onto unit")
2500                 End If
2501             End If
2502
2503             Return unit
2504
2505         End Function
2506
2507         Public Function GetUnitTime(ByVal worker As BackgroundWorker, ByVal e As DoWorkEventArgs) As    ↵
           ReferenceTime
2508             'Get the unit time and the PC time to check the time difference
2509             'take 8 samples, then average
2510             'The average represents a snapshot of the measured unit and computer time, taken
2511             '  at a single point in time
2512             'The averaging is done solely to reduce the influence of communications latency on the difference
2513             '  between the recorded unit time and the recorded computer time
2514
2515             'Note: unit time is returned in the format "Wed 28Jun2006 13:39:15" (with ends trimmed)
2516
2517             Dim result As String = ""
2518             Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab)}
2519             Dim tDifferenceTemp(7) As Double
2520             Dim unitDateTicks As Long = 0
2521             Dim hostDateTicks As Long = 0
2522             Dim unitDate(7) As Date
2523             Dim hostDate(7) As Date
2524             Dim tOffsetAvg As Double = 0
2525             Dim tryp() As Boolean = {False, False, False, False, False, False, False, False}
2526             Dim proceed As Boolean = True
2527
2528             'take 4 samples
2529             For i As Integer = 0 To 3
2530                 Dim tDifferenceTemp2 As TimeSpan = New TimeSpan(0)
2531
2532                 'unitdate(i) = New Date(1)
2533
2534                 'The host time is captured before the unit time is received, because Ed has observed that the
2535                 'sending latency of the command sent to the 870 is less than the receiving latency of the    ↵
           response
2536                 'sent back to the computer
2537
2538
2539                 Write("R2" & vbCr)
2540                 hostDate(i) = Date.Now       'immediately get computer time
2541                 result = ReadLine(CChar(vbLf), 5) 'read host time
2542
2543
2544
2545                 result = result.Trim(trimChars)
2546
2547                 If result Like "*No response.*" OrElse result.Length > 30 Then   'readline read an invalid date  ↵
           string
2548                     tryp(i) = False
2549                     Exit For 'and checker will get the time again
2550                 ElseIf result Like "*\?\?\?*" Then
2551                     'no date is entered - enter today's date and day of week
2552                     Write("M4" & vbCr)
2553                     ReadLine(CChar(vbLf))
2554                     Write("S7," & Date.Now.ToString("MM/dd/yy") & vbCr)
2555                     ReadLine(CChar(vbLf))
2556                     Write("S8," & Date.Now.DayOfWeek + 1 & vbCr)
2557                     ReadLine(CChar(vbLf))
2558                     Write("M3" & vbCr)
2559                     ReadLine(CChar(vbLf))
2560                 ElseIf Not result Like "#*" Then
2561                     result = result.Remove(0, 4)    'trim off the day of week from the date - it is redundant
2562                     '                                     and will introduce errors if it was set wrong on the 870
```

```vb
2563                End If
2564
2565                tryp(i) = Date.TryParse(result, unitDate(i))
2566                hostDate(i) = hostDate(i).AddMilliseconds(45)    'assume 45 msec transmission delay -
2567                'this is based on the delay for the NIST ACTS timekeeping service
2568
2569                tDifferenceTemp2 = unitDate(i).Subtract(hostDate(i))
2570                tDifferenceTemp(i) = tDifferenceTemp2.TotalSeconds 'Mod 3600    'ignore the difference in hours ↙
        -
2571                '                                                prevents time zone differences from
2572                '                                                screwing up the recorded time
2573
2574         Next
2575
2576         'wait .5 secs
2577         Thread.Sleep(500)
2578
2579         'take 4 more samples
2580         For i As Integer = 4 To 7
2581             Dim tDifferenceTemp2 As TimeSpan = New TimeSpan(0)
2582
2583             'unitDate(i) = New Date(1)
2584
2585             'The host time is captured before the unit time is received, because Ed has observed that the
2586             'sending latency of the command sent to the 870 is less than the receiving latency of the ↙
        response
2587             'sent back to the computer
2588
2589
2590             Write("R2" & vbCr)
2591             hostDate(i) = Date.Now        'immediately get computer time
2592             result = ReadLine(CChar(vbLf), 5) 'read host time
2593
2594
2595
2596             result = result.Trim(trimChars)
2597             If result Like "*No response.*" OrElse result.Length > 30 Then
2598                 tryp(i) = False
2599                 Exit For 'and checker will get the time again
2600             ElseIf Not result Like "#*" Then
2601                 result = result.Remove(0, 4)    'trim off the day of week from the date - it is redundant
2602                 '                                and will introduce errors if it was set wrong on the 870
2603             End If
2604
2605             'UnitDate(i) is passed to tryparse by reference, and thus is edited directly by this method
2606             tryp(i) = Date.TryParse(result, unitDate(i))
2607             hostDate(i) = hostDate(i).AddMilliseconds(45)    'assume 45 msec transmission delay -
2608             'this is based on the delay for the NIST ACTS timekeeping service
2609
2610             tDifferenceTemp2 = unitDate(i).Subtract(hostDate(i))
2611             tDifferenceTemp(i) = tDifferenceTemp2.TotalSeconds 'Mod 3600    'ignore the difference in hours ↙
        -
2612             '                                                prevents time zone differences from
2613             '                                                screwing up the recorded time
2614         Next
2615
2616         'find the average time and offset for the first three trials
2617         'shifting bits to the right by 3 is the same as dividing by 8
2618
2619         For i As Integer = 0 To 7
2620             unitDateTicks += unitDate(i).Ticks >> 3
2621             hostDateTicks += hostDate(i).Ticks >> 3
2622             tOffsetAvg += tDifferenceTemp(i) * 0.125
2623         Next
2624
2625         Dim RT As New ReferenceTime(New Date(unitDateTicks), New Date(hostDateTicks), tOffsetAvg)
2626
2627         'check values
2628         'MessageBox.Show(RT.UnitTimeZero & vbLf & RT.HostTimeZero, "Time Check")
2629
2630         'Sometimes the 870 will not receive a sent command, or the program will time out before
2631         'the 870's response is received. This condition will result in only 7 responses received
2632         'for the 8 commands sent. This routine needs to verify that 8 responses were received
2633         'so that it doesn't return an errorneous unit time value.
2634
2635         'if less than 8 responses were received, the tOffsetAvg will be very large, on the order of years
2636         '(1 year = 31557600 seconds)
2637
2638         For i As Integer = 0 To 7
2639             'if any tryp(i) is false, get time again
2640             If tryp(i) = False Then
2641                 proceed = False
2642             End If
2643         Next
2644
2645         'let user break this loop, in case the program gets stuck (the corrupted RT won't be saved)
2646         If worker.CancellationPending Then
2647             e.Cancel = True
2648             Write("M12" & vbCr)
2649             result = ReadLine(CChar(vbLf))
2650             Throw New CancelException()
2651         End If
```

```vb
2652
2653            If tOffsetAvg > 31557600 OrElse proceed = False Then
2654                Notify("." & vbLf)
2655                Me.getUnitTimeCounter += 1
2656                If Me.getUnitTimeCounter > 10 Then
2657                    Me.getUnitTimeCounter = 0
2658                    Throw New FormatException("Can't read from unit - will try again later")
2659                End If
2660                RT = GetUnitTime(worker, e)  'run until RT is correct - last RT will be returned to the calling ↙
        program
2661            End If
2662
2663            Return RT
2664
2665        End Function
2666
2667        Private Shared Function IsE(ByVal line As String) As Boolean
2668            If line.StartsWith("E") AndAlso line.Length >= 153 AndAlso NumCommas(line) = 18 Then
2669                'try to parse each cell - takes lots of CPU time,
2670                'but is needed to ensure the data will go into the database smoothly
2671                'note that the corrected time has not been appended to these records yet - they will be 19 cols ↙
        wide
2672                Dim sep() As Char = {","c}
2673                Dim cells() As String = line.Split(sep, StringSplitOptions.None)
2674                Dim tryp(cells.Length - 1) As Boolean
2675                Dim trypFinal As Boolean = True
2676                Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab), "-"c}
2677
2678                For i As Integer = 0 To cells.Length - 1
2679                    cells(i) = cells(i).Trim(trimChars)
2680                Next
2681                tryp(0) = True
2682                For i As Integer = 1 To tryp.Length - 1
2683                    tryp(i) = False
2684                Next
2685                tryp(1) = Date.TryParse(cells(1), New Date)
2686                If cells(2).Length < 11 Then
2687                    tryp(2) = True
2688                End If
2689                For i As Integer = 3 To 7
2690                    tryp(i) = Double.TryParse(cells(i), New Double)
2691                Next
2692                For i As Integer = 8 To 9
2693                    tryp(i) = Integer.TryParse(cells(i), New Integer)
2694                Next
2695                If cells(10).Length < 16 Then
2696                    tryp(10) = True
2697                End If
2698                For i As Integer = 11 To 13
2699                    tryp(i) = Double.TryParse(cells(i), New Double)
2700                Next
2701                If cells(14).Length < 4 Then
2702                    tryp(14) = True
2703                End If
2704                For i As Integer = 15 To 18
2705                    tryp(i) = Double.TryParse(cells(i), New Double)
2706                Next
2707
2708                For i As Integer = 0 To tryp.Length - 1
2709                    If tryp(i) = False Then
2710                        trypFinal = False
2711                    End If
2712                Next
2713
2714                If trypFinal = True Then
2715                    Return True
2716                Else
2717                    Return False
2718                End If
2719            Else
2720                Return False
2721            End If
2722        End Function
2723        Private Shared Function IsI(ByVal line As String) As Boolean
2724            If line.StartsWith("I") AndAlso line.Length >= 237 AndAlso NumCommas(line) = 33 Then
2725                'try to parse each cell - takes lots of CPU time,
2726                'but is needed to ensure the data will go into the database smoothly
2727                'note that the corrected time has not been appended to these records yet - they will be 34 cols ↙
        wide
2728                Dim sep() As Char = {","c}
2729                Dim cells() As String = line.Split(sep, StringSplitOptions.None)
2730                Dim tryp(cells.Length - 1) As Boolean
2731                Dim trypFinal As Boolean = True
2732                Dim trimChars As Char() = {CChar(vbCr), CChar(vbLf), "%"c, " "c, CChar(vbTab), "-"c}
2733
2734                For i As Integer = 0 To cells.Length - 1
2735                    cells(i) = cells(i).Trim(trimChars)
2736                Next
2737                tryp(0) = True
2738                For i As Integer = 1 To tryp.Length - 1
2739                    tryp(i) = False
2740                Next
```

```vb
2741                    tryp(1) = Date.TryParse(cells(1), New Date)
2742                    If cells(2).Length < 11 Then
2743                        tryp(2) = True
2744                    End If
2745                    For i As Integer = 3 To 8
2746                        tryp(i) = Double.TryParse(cells(i), New Double)
2747                    Next
2748                    For i As Integer = 9 To 13
2749                        tryp(i) = Integer.TryParse(cells(i), New Integer)
2750                    Next
2751                    tryp(14) = Double.TryParse(cells(14), New Double)
2752                    tryp(15) = Integer.TryParse(cells(15), New Integer)
2753                    tryp(16) = Double.TryParse(cells(16), New Double)
2754                    tryp(17) = Integer.TryParse(cells(17), New Integer)
2755                    tryp(18) = Double.TryParse(cells(18), New Double)
2756                    tryp(19) = Integer.TryParse(cells(19), New Integer)
2757                    tryp(20) = Double.TryParse(cells(20), New Double)
2758                    tryp(21) = Integer.TryParse(cells(21), New Integer)
2759                    tryp(22) = Double.TryParse(cells(22), New Double)
2760                    tryp(23) = Integer.TryParse(cells(23), New Integer)
2761                    For i As Integer = 24 To 26
2762                        tryp(i) = Double.TryParse(cells(i), New Double)
2763                    Next
2764                    If cells(27).Length < 4 Then
2765                        tryp(27) = True
2766                    End If
2767                    For i As Integer = 28 To 33
2768                        tryp(i) = Double.TryParse(cells(i), New Double)
2769                    Next
2770
2771                    For i As Integer = 0 To tryp.Length - 1
2772                        If tryp(i) = False Then
2773                            trypFinal = False
2774                        End If
2775                    Next
2776                    If trypFinal = True Then
2777                        Return True
2778                    Else
2779                        Return False
2780                    End If
2781                Else
2782                    Return False
2783                End If
2784            End Function
2785            Private Shared Function IsR(ByVal line As String) As Boolean
2786                If line.StartsWith("R") AndAlso NumCommas(line) > 0 Then
2787                    Return True
2788                Else
2789                    Return False
2790                End If
2791            End Function
2792            Private Shared Function IsQ(ByVal line As String) As Boolean
2793                If line.StartsWith("Q") AndAlso NumCommas(line) > 0 Then
2794                    Return True
2795                Else
2796                    Return False
2797                End If
2798            End Function
2799            Private Shared Function IsL(ByVal line As String) As Boolean
2800                If line.StartsWith("L") AndAlso NumCommas(line) = 5 Then
2801                    Return True
2802                Else
2803                    Return False
2804                End If
2805            End Function
2806            Private Shared Function IsC(ByVal line As String) As Boolean
2807                If line.StartsWith("C") AndAlso NumCommas(line) = 4 Then
2808                    Return True
2809                Else
2810                    Return False
2811                End If
2812            End Function
2813            Private Shared Function NumCommas(ByVal line As String) As Integer
2814
2815                Dim re As New Regex(",")
2816                Dim mc As MatchCollection = re.Matches(line)
2817
2818                Return mc.Count
2819            End Function
2820            Private Shared Function IsBadEntry(ByVal line As String) As Boolean
2821                'tests to see if line is a bad entry
2822
2823                If line.Length < 2 Then        'blank line
2824                    Return False
2825                Else
2826                    If (line.StartsWith("E") AndAlso NumCommas(line) = 18) OrElse _
2827                    (line.StartsWith("I") AndAlso NumCommas(line) = 33) OrElse _
2828                    (line.StartsWith("R") AndAlso NumCommas(line) > 0) OrElse _
2829                    (line.StartsWith("Q") AndAlso NumCommas(line) > 0) OrElse _
2830                    (line.StartsWith("L") AndAlso NumCommas(line) = 5) OrElse _
2831                    (line.StartsWith("C") AndAlso NumCommas(line) = 4) Then
2832                        Return False
```

```vb
2833                 Else        'line wasn't added to one of the lists
2834                     Return True
2835                 End If
2836         End If
2837     End Function
2838
2839 #End Region
2840
2841 #Region "Junk Methods and Vars"
2842
2843
2844     'Public Sub Reconfigure(ByVal lnum As String, ByVal worker As BackgroundWorker, ByVal e As       ↙
         DoWorkEventArgs)
2845     '     'Dials a unit to reconfigure it only
2846
2847     '     If modemMode = Mode.Listen Then
2848     '         Exit Sub
2849     '     End If
2850
2851     '     Dim result As String = ""
2852     '     Dim deleg As New GetUnitDelegate(AddressOf Form1.GetUnit)
2853     '     Dim deleg2 As New ReturnUnitDelegate(AddressOf Form1.ReturnUnit)
2854     '     Dim unit As LDUnit = deleg.Invoke(lnum)
2855
2856
2857     '     If String.Compare(unit.UnitNum, "null") = 0 Then
2858     '         Exit Sub
2859     '     End If
2860
2861     '     Notify("*** Now dialing unit " & unit.UnitNum & " ***" & vbCr)
2862
2863     '     Try
2864     '         If comx.IsOpen Then
2865     '             comx.Close()
2866     '         End If
2867     '         comx.Open()
2868     '         comx.DiscardInBuffer()  'flush receive buffer
2869
2870     '         Write("AT" & vbCr)
2871     '         result = ReadLine(CChar(vbLf))
2872     '         result &= ReadLine(CChar(vbLf))
2873     '         Write("ATZ" & vbCr)
2874     '         result = ReadLine(CChar(vbLf))
2875     '         result &= ReadLine(CChar(vbLf))
2876     '         'MessageBox.Show(result, "result after first write")
2877     '         If Not result Like "*OK*" Then
2878     '             comx.DiscardInBuffer()
2879     '             Write("AT" & vbCr)
2880     '             result = ReadLine(CChar(vbLf))
2881     '             result &= ReadLine(CChar(vbLf))
2882     '             'MessageBox.Show(result, "result after second AT command")
2883     '             If Not result Like "*OK*" Then
2884     '                 Throw New NoModemException("Modem on " & comx.PortName & " not present or responsive")
2885     '             End If
2886     '         End If
2887     '         Write("ATDT" & unit.UnitPhoneNum & vbCr)
2888     '         result = ReadLine(CChar(vbLf), 90)
2889     '         result &= ReadLine(CChar(vbLf), 90)
2890     '         If result Like "*OK*" Then
2891     '             result &= ReadLine(CChar(vbLf), 90)
2892     '         End If
2893     '         If Not result Like "*CONNECT*" Then
2894     '             Write("atz" & vbCr)
2895     '             result = ReadLine(CChar(vbLf))
2896     '             result &= ReadLine(CChar(vbLf))
2897     '             Write("ATDT" & unit.UnitPhoneNum & vbCr)
2898     '             result = ReadLine(CChar(vbLf), 90)
2899     '             result &= ReadLine(CChar(vbLf), 90)
2900     '             If Not result Like "*CONNECT*" Then
2901     '                 Throw New NoConnectException("Unable to connect to remote modem")
2902     '             End If
2903     '         End If
2904
2905     '         If unit.UnitOwner = LDUnit.Owner.CERL Then  'limit modem speed to 14400 bps
2906     '             If comx.PortName = "COM1" Then
2907     '                 Write("AT$MB14400 \N3" & vbCr)
2908     '                 result = ReadLine(CChar(vbLf))
2909     '                 result &= ReadLine(CChar(vbLf))
2910     '             Else
2911     '                 Write("AT+MS=V32B,1,300,14400,300,14400" & vbCr)
2912     '                 result = ReadLine(CChar(vbLf))
2913     '                 result &= ReadLine(CChar(vbLf))
2914     '                 Write("AT\N3" & vbCr)
2915     '                 result = ReadLine(CChar(vbLf))
2916     '                 result &= ReadLine(CChar(vbLf))
2917     '             End If
2918     '         End If
2919
2920     '         unit = getPass(unit, worker, e)
2921     '         If unit.UnitNum = "null" Then
2922     '             Exit Sub
2923     '         End If
```

```
2924    '
2925    '            ReconfigureUnit(unit)   'reconfigures unit on same thread
2926    '
2927    '            Write("M12" & vbCr)
2928    '            result = ReadLine(CChar(vbLf), 10)
2929    '            'result &= ReadLine(CChar(vbLf), 10)
2930    '            Write("+++ath" & vbCr)
2931    '            result = ReadLine(CChar(vbLf))
2932    '            result &= ReadLine(CChar(vbLf))
2933    '
2934    '            comx.Close()
2935    '
2936    '        Catch ex As NoModemException     'don't use this modem to download the remaining units
2937    '            If comx.IsOpen Then
2938    '                comx.Close()
2939    '            End If
2940    '            Notify("Error: " & ex.msg & vbLf)
2941    '            unit.IsDownloadInProgress = False
2942    '            unit.IsDownloadDone = False
2943    '            deleg2.Invoke(unit)
2944    '            Exit Sub
2945    '        Catch ex As NoConnectException      'what happens when the line is busy or unavailable
2946    '            If comx.IsOpen Then
2947    '                comx.Close()
2948    '            End If
2949    '            Notify("Error: " & ex.msg & vbLf)
2950    '            unit.IsDownloadInProgress = False
2951    '            unit.IsDownloadDone = False
2952    '            unit.DialTries += 1
2953    '            If unit.DialTries > 6 Then
2954    '                'stop trying to download unit
2955    '                unit.IsDownloadDone = True
2956    '                Notify("Attempted to dial unit " & unit.UnitNum & " seven times." & vbLf)
2957    '                Notify("Will not try again until the next Dial command is issued." & vbLf)
2958    '            End If
2959    '            deleg2.Invoke(unit)
2960    '            Dial(worker, e)
2961    '        Catch ex As CancelException     'user cancels the operation - close the port and exit
2962    '            Notify(ex.msg & vbLf)
2963    '            If comx.IsOpen Then
2964    '                comx.Close()
2965    '            End If
2966    '            unit.IsDownloadInProgress = False
2967    '            unit.IsDownloadDone = False
2968    '            deleg2.Invoke(unit)
2969    '            e.Cancel = True
2970    '            Exit Sub
2971    '        Catch ex As InvalidOperationException      'port is closed when attempting to write data
2972    '            Notify("InvalidOperationException" & vbLf)
2973    '            Notify("Error: " & ex.Message & vbLf)
2974    '            If comx.IsOpen Then
2975    '                comx.Close()
2976    '            End If
2977    '            unit.IsDownloadInProgress = False
2978    '            unit.IsDownloadDone = False
2979    '            deleg2.Invoke(unit)
2980    '            Dial(worker, e)
2981    '        Catch ex As IO.IOException       'port is closed when attempting to write data
2982    '            Notify("IOException" & vbLf)
2983    '            Notify("Error: " & ex.Message & vbLf)
2984    '            If comx.IsOpen Then
2985    '                comx.Close()
2986    '            End If
2987    '            unit.IsDownloadInProgress = False
2988    '            unit.IsDownloadDone = False
2989    '            deleg2.Invoke(unit)
2990    '            Dial(worker, e)
2991    '        Catch ex As Exception
2992    '            'will usually be from the download routine, and result from a dropped
2993    '            'or error-prone connection
2994    '
2995    '            Write("M12" & vbCr)       'tell 870 to shut down
2996    '            result = ReadLine(CChar(vbLf), 10)
2997    '            'result &= ReadLine(CChar(vbLf), 10)
2998    '            Write("+++ath" & vbCr)
2999    '            result = ReadLine(CChar(vbLf))
3000    '            result &= ReadLine(CChar(vbLf))
3001    '            If comx.IsOpen Then
3002    '                comx.Close()
3003    '            End If
3004    '            Notify("Error: " & ex.Message & vbLf)
3005    '            unit.IsDownloadInProgress = False
3006    '            unit.IsDownloadDone = False
3007    '            unit.DialTries += 1
3008    '            If unit.DialTries > 6 Then
3009    '                'stop trying to download unit
3010    '                unit.IsDownloadDone = True
3011    '                Notify("Attempted to dial unit " & unit.UnitNum & " seven times." & vbLf)
3012    '                Notify("Will not try again until the next Dial command is issued." & vbLf)
3013    '            End If
3014    '            deleg2.Invoke(unit)
3015    '        End Try
```

```vb
3016
3017     'End Sub
3018
3019
3020
3021
3022
3023     'Public Sub UpdateBuffer() 'ByVal value As String)
3024     '    rxBuffer.Append(comx.ReadExisting) 'value)
3025
3026     'End Sub
3027
3028     'Public Delegate Sub UpdateBufferDelegate() 'ByVal value As String)
3029
3030     'Public Sub ReadPort(ByVal sender As Object, ByVal e As SerialDataReceivedEventArgs)
3031     '    'This handler will be added and removed at will by the program
3032
3033     '    'The DataReceived event fires when the data in the serial port buffer exceeds the
3034     '    'ReceivedBytesThreshold value
3035     '    'The default ReceivedBytesThreshold value is 1 byte
3036
3037     '    Dim deleg As UpdateBufferDelegate
3038
3039     '    deleg = New UpdateBufferDelegate(AddressOf UpdateBuffer)
3040     '    deleg.Invoke() 'comx.ReadExisting
3041     'End Sub
3042
3043     'Public Function ReadUntilChar(ByVal searchChar As Char) As String
3044     '    'reads port until searchChar (usually vbLf) is found
3045     '    'returns the string of chars in the rxBuffer between the time this method was invoked
3046     '    'and searchChar was found
3047     '    'if timeout exceeded, just returns the characters received since this method was invoked,
3048     '    'and lets the calling method figure out what went wrong
3049     '    'Only use when ReadLine doesn't return a value
3050
3051     '    Dim index As Integer = -1    'start index
3052     '    Dim index2 As Integer = -1   'end index
3053     '    Dim intermediateIndex As Integer = -1
3054     '    Dim countSinceLastCharReceived As Integer = 0  'timeout mechanism
3055     '    Dim lastChar As Char = "|"c
3056     '    Dim returnSB As New StringBuilder(100)
3057     '    Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
3058     '        New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
3059
3060
3061     '    rxBuffer.Append("<br>")    'delineates a new command will be sent, prevents method from returning ↙
         a false positive
3062     '    index = rxBuffer.Length - 1 'because the last index is always length-1
3063     '    intermediateIndex = index
3064
3065     '    'For debugging
3066     '    'MessageBox.Show(rxBuffer.ToString, "rxBuffer")
3067
3068     '    'loop until the search char is found or until no new data has been received for the timeout count
3069     '    Do Until lastChar = searchChar OrElse countSinceLastCharReceived >= 300  '3 secs
3070     '        index2 = rxBuffer.Length - 1
3071     '        If intermediateIndex <> index2 Then
3072     '            intermediateIndex = index2
3073     '            countSinceLastCharReceived = 0  'reset counter if new data in rxbuffer
3074     '        Else
3075     '            countSinceLastCharReceived += 1 'increment counter if no new data
3076     '        End If
3077     '        lastChar = rxBuffer.Chars(index2)
3078     '        Threading.Thread.Sleep(10)  'pause for 10 ms - prevents this loop from hogging the computer
3079     '    Loop
3080     '    If index + 1 <= index2 Then      'data was received
3081     '        For i As Integer = index + 1 To index2
3082     '            returnSB.Append(rxBuffer.Chars(i))
3083     '        Next
3084     '    End If
3085
3086     '    'update text box
3087     '    EA.AppendThisString = returnSB.ToString
3088     '    RaiseEvent UpdateOutBoxesEvent(Me, EA)
3089
3090     '    Return returnSB.ToString
3091
3092     'End Function
3093
3094     'Public Function ReadUntilChar(ByVal searchChar As Char, ByVal timeout As Double) As String
3095     '    'Overloads ReadUntilChar, lets calling method set a timeout in seconds
3096     '    '(e.g. for connecting to a host modem)
3097
3098
3099     '    Dim index As Integer = -1    'start index
3100     '    Dim index2 As Integer = -1   'end index
3101     '    Dim intermediateIndex As Integer = -1
3102     '    Dim countSinceLastCharReceived As Integer = 0  'timeout mechanism
3103     '    Dim lastChar As Char = "|"c
3104     '    Dim returnSB As New StringBuilder(100)
3105     '    Dim time As Integer = CInt(timeout * 100)
3106     '    Dim EA As WindowsApplication1.UpdateOutBoxesEventArgs = _
```

```
3107     '          New UpdateOutBoxesEventArgs("", UpdateOutBoxesEventArgs.RW.Read)
3108
3109     '    rxBuffer.Append("<br>")      'delineates a new command will be sent, prevents method from returning ⤸
         a false positive
3110     '    index = rxBuffer.Length - 1 'because the last index is always length-1
3111     '    intermediateIndex = index
3112
3113     '    'For debugging
3114     '    'MessageBox.Show(rxBuffer.ToString, "rxBuffer")
3115
3116     '    'loop until the search char is found or until no new data has been received for the timeout count
3117     '    Do Until lastChar = searchChar OrElse countSinceLastCharReceived >= time
3118     '        index2 = rxBuffer.Length - 1
3119     '        If intermediateIndex <> index2 Then
3120     '            intermediateIndex = index2
3121     '            countSinceLastCharReceived = 0  'reset counter if new data in rxbuffer
3122     '        Else
3123     '            countSinceLastCharReceived += 1 'increment counter if no new data
3124     '        End If
3125     '        lastChar = rxBuffer.Chars(index2)
3126     '        Threading.Thread.Sleep(10)   'pause for 10 ms - prevents this loop from hogging the computer
3127     '    Loop
3128
3129     '    For i As Integer = index + 1 To index2
3130     '        returnSB.Append(rxBuffer.Chars(i))
3131     '    Next
3132
3133     '    'update text box
3134     '    EA.AppendThisString = returnSB.ToString
3135     '    RaiseEvent UpdateOutBoxesEvent(Me, EA)
3136     '    Return returnSB.ToString
3137
3138     'End Function
3139
3140
3141
3142
3143
3144
3145
3146
3147     'Private timeOut As Integer = 1
3148     'Private timeOutCountUp As Boolean = True
3149     'Private newData As Integer = 1
3150     'Private newDataCountUp As Boolean = True
3151     'Public WithEvents timeoutTimer As Timers.Timer = New Timers.Timer(500) 'fire an event every 500 msec
3152     'Private txtOutBox As New Windows.Forms.TextBox
3153
3154     'Public Sub TimeOutHandler(ByVal sender As Object, ByVal e As Timers.ElapsedEventArgs) _
3155     '    Handles timeoutTimer.Elapsed
3156     '    'this runs on a new thread - need to delegate the increment task back to the Modem thread
3157     '    Dim deleg As IncrementTimeOutDelegate = New IncrementTimeOutDelegate(AddressOf IncrementTimeOut)
3158     '    deleg.Invoke()
3159     'End Sub
3160
3161     'Public Delegate Sub IncrementNewDataDelegate()
3162     'Public Delegate Sub IncrementTimeOutDelegate()
3163
3164     'Public Sub IncrementTimeOut()
3165     '    If timeOut <= 0 Then
3166     '        timeOutCountUp = True
3167     '    ElseIf timeOut > 1500000000 Then
3168     '        timeOutCountUp = False      'count down
3169     '    End If
3170
3171     '    If timeOutCountUp = True Then   'increment
3172     '        timeOut += 1
3173     '    Else                            'decrement
3174     '        timeOut = timeOut - 1
3175     '    End If
3176     'End Sub
3177
3178
3179     'Public Sub IncrementNewData()
3180     '    If newData <= 0 Then
3181     '        newDataCountUp = True
3182     '    ElseIf newData >= 1500000000 Then
3183     '        newDataCountUp = False      'count down
3184     '    End If
3185
3186     '    If newDataCountUp Then   'increment
3187     '        newData += 1
3188     '    Else                            'decrement
3189     '        newData = newData - 1
3190     '    End If
3191     'End Sub
3192
3193     'Public Function SendCommand(ByVal writeString As String, ByVal readTimeOut As Integer) As Integer
3194     '    'halts the calling method's execution until data is available at the serial port
3195     '    'if no data was found before specified timeout, returns false
3196
3197
```

```
3198    '    'Do Until localND <> newData OrElse localTO < timeOut - readTimeOut _
3199    '    '  OrElse localTO > timeOut + readTimeOut
3200    '    '    Threading.Thread.Sleep(25)  'pause 25 msec - keeps loop from eating all the processor time
3201    '    'Loop
3202    '    'Return False
3203
3204    '    Dim index1 As Integer = rxBuffer.Length    'get current end position of receive buffer
3205    '    Dim index2 As Integer = 0
3206    '    'Dim localTO As Long = timeOut
3207    '    'Dim localND As Long = newData
3208
3209    '    Write(writeString)
3210
3211    '    'wait readTimeOut after the last data character is received
3212
3213    '    Dim lastChar As Char = rxBuffer.Chars(rxBuffer.Length)
3214
3215    'End Function
3216
3217    'Public Function ReadUntil(ByVal searchPattern As String) As Boolean
3218    '    'reads input until the search string is found
3219    '    'use for procedural control when talking to the remote unit
3220    '    'possibly overload with a string array input parameter
3221    '    'return True if the string was found, false if string was not found
3222    '    'make sure to quit after a specified timeout
3223
3224    '    Dim found As Boolean = False
3225    '    Dim isMoreText As Boolean = True
3226
3227    '    rxBuffer.Remove(0, rxBuffer.Length)    'clear buffer
3228
3229    '    Do While isMoreText
3230    '        '    isMoreText = Read(2)    '1 second timeout for more data to be received
3231    '        If Not isMoreText Then
3232    '            Exit Do
3233    '        End If
3234    '    Loop
3235
3236    '    If rxBuffer.Length = 0 Then    'nothing read
3237    '        Return False
3238    '    End If
3239
3240    '    found = rxBuffer.ToString Like searchPattern
3241    '    Return found
3242
3243    'End Function
3244
3245
3246
3247
3248
3249    'Dim msgBoxResult As Windows.Forms.DialogResult = _
3250    'MessageBox.Show("The following records from unit " & unit.UnitNum & " are invalid:" _
3251    '& vbCr & vbCr & badListStr & vbCr & vbCr & _
3252    '"These records have been written to the file " & unit.InstallPath & _
3253    '"Bad Records\error" & unit.UnitNum & " - " & _
3254    'hostDateAvg.ToLongDateString & ".txt" & vbCr & vbCr & _
3255    '"Abort: Reset the unit's memory and reconfigure report settings" & vbCr & _
3256    '"Retry: Try download again" & vbCr & _
3257    '"Ignore: Remove bad records from the report and continue", _
3258    '"Warning", _
3259    'MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button2)
3260
3261    'Select Case msgBoxResult
3262    '    Case DialogResult.Abort
3263
3264    '        Write("R1" & vbCr)
3265    '        result = ReadLine(CChar(vbLf))
3266    '        If result Like "*Larson*" Then
3267    '            'Write("M4" & vbCr)
3268    '            'result = ReadLine(CChar(vbLf))
3269    '            'Write("S1,1" & vbCr)
3270    '            'result = ReadLine(CChar(vbLf))
3271    '            'Write("M3" & vbCr)
3272    '            'result = ReadLine(CChar(vbLf))
3273    '            ReconfigureUnit(unit)
3274    '        Else
3275    '            unit.IsDownloadInProgress = False
3276    '            unit.IsDownloadDone = False
3277    '            Notify("Warning! Unit " & unit.UnitNum & " has timed out." & vbCr)
3278    '            Notify("Download will resume after the other units have finished." & vbCr)
3279
3280    '            'return unit to the LDUnitList,
3281    '            deleg2.Invoke(unit)
3282    '            Exit Sub
3283    '        End If
3284    '        downloadAgain = True
3285
3286    '    Case DialogResult.Retry
3287    '        downloadAgain = True
3288    '    Case DialogResult.Ignore
3289    '        Exit Select
```

```
3290        'End Select
3291
3292
3293
3294 #End Region
3295
3296 End Class
3297
3298 Public Class NoModemException
3299     Inherits ApplicationException
3300
3301     Protected m_msg As String
3302
3303     Public Property msg() As String
3304         Get
3305             Return m_msg
3306         End Get
3307         Set(ByVal value As String)
3308             m_msg = value
3309         End Set
3310     End Property
3311
3312     Public Sub New(ByVal message As String)
3313         msg = message
3314     End Sub
3315
3316 End Class
3317
3318 Public Class NoConnectException
3319     Inherits ApplicationException
3320
3321     Protected m_msg As String
3322
3323     Public Property msg() As String
3324         Get
3325             Return m_msg
3326         End Get
3327         Set(ByVal value As String)
3328             m_msg = value
3329         End Set
3330     End Property
3331
3332     Public Sub New(ByVal message As String)
3333         msg = message
3334     End Sub
3335
3336 End Class
3337
3338 Public Class CancelException
3339     Inherits ApplicationException
3340     'thrown when user cancels an operation
3341
3342     Protected m_msg As String
3343
3344     Public Property msg() As String
3345         Get
3346             Return m_msg
3347         End Get
3348         Set(ByVal value As String)
3349             m_msg = value
3350         End Set
3351     End Property
3352
3353     Public Sub New()
3354         msg = "Canceled by user."
3355     End Sub
3356
3357     Public Sub New(ByVal message As String)
3358         msg = message
3359     End Sub
3360
3361 End Class
3362
3363 Public Class NotEnabledException
3364     Inherits ApplicationException
3365     'thrown when user cancels an operation
3366
3367     Protected m_msg As String
3368
3369     Public Property msg() As String
3370         Get
3371             Return m_msg
3372         End Get
3373         Set(ByVal value As String)
3374             m_msg = value
3375         End Set
3376     End Property
3377
3378     Public Sub New()
3379         msg = "Unit is not enabled."
3380     End Sub
3381
```

```vb
3382     Public Sub New(ByVal message As String)
3383          msg = message
3384     End Sub
3385
3386 End Class
```

Appendix D: Dial One Unit

```vb
1  Imports System.Windows.Forms
2
3  Public Class OneUnitDialog
4
5      Private ulist As List(Of LDUnit)
6
7      Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK_Button↵
       .Click
8          Me.DialogResult = System.Windows.Forms.DialogResult.OK
9
10         'to only download one unit, mark all but the selected unit as having been downloaded
11         If Cb1.SelectedIndex = 0 Then
12             For Each unit As LDUnit In ulist
13                 unit.IsDownloadDone = False
14                 ci(unit)
15             Next
16         ElseIf Cb1.SelectedIndex = 1 Then
17             For Each unit As LDUnit In ulist
18                 If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
19                     unit.IsDownloadDone = False
20                     ci(unit)
21                 Else
22                     unit.IsDownloadDone = True
23                     ci(unit)
24                 End If
25             Next
26         ElseIf Cb1.SelectedIndex = 2 Then
27             For Each unit As LDUnit In ulist
28                 If unit.UnitOwner = LDUnit.Owner.CERL Then
29                     unit.IsDownloadDone = False
30                     ci(unit)
31                 Else
32                     unit.IsDownloadDone = True
33                     ci(unit)
34                 End If
35             Next
36         Else
37             Dim u As LDUnit = ulist.Item(Cb1.SelectedIndex - 3)
38             ulist.RemoveAt(Cb1.SelectedIndex - 3)
39
40             For Each unit As LDUnit In ulist
41                 unit.IsDownloadDone = True
42             Next
43
44             u.IsDownloadDone = False
45             ci(u)
46             ulist.Add(u)
47         End If
48
49         Me.Close()
50
51      End Sub
52
53      Private Sub ci(ByRef unit As LDUnit)
54          If CheckBox1.Checked Then
55              unit.Initialize = True
56          Else
57              unit.Initialize = False
58          End If
59      End Sub
60
61      Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles       ↵
       Cancel_Button.Click
62          Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
63          Me.Close()
64      End Sub
65
66      Private Sub OneUnitDialog_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.            ↵
       FormClosingEventArgs) Handles Me.FormClosing
67          Form1.ReturnAllUnits(ulist)
68
69      End Sub
70
71      Private Sub OneUnitDialog_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase↵
       .Load
72          Dim cb1List As New ComboBox.ObjectCollection(Cb1)
73          Dim ucomp As New uCompare()
74          Dim comp As Collections.Generic.IComparer(Of LDUnit) = ucomp
75
76
77          ulist = Form1.GetAllUnitsNoMark()
78
79          Cb1.Items.Add("All Units")
80          Cb1.Items.Add("Aberdeen")
81          Cb1.Items.Add("CERL")
82
83          ulist.Sort(comp)
84
85          For Each unit As LDUnit In ulist
86              Cb1.Items.Add(unit.UnitNum)
87          Next
88
```

```vb
 89
 90         End Sub
 91  End Class
 92
 93  Public Class uCompare
 94         Implements Collections.Generic.IComparer(Of LDUnit)
 95
 96         Public Function Compare(ByVal x As LDUnit, ByVal y As LDUnit) As Integer Implements System.Collections.  ↵
       Generic.IComparer(Of LDUnit).Compare
 97             Dim unit1 As LDUnit = DirectCast(x, LDUnit)
 98             Dim unit2 As LDUnit = DirectCast(y, LDUnit)
 99
100             Return String.Compare(unit1.UnitNum, unit2.UnitNum)
101         End Function
102  End Class
```

Appendix E: Unit Options

```vb
1  Public Class UnitOptions
2
3      Private workingUList As New List(Of LDUnit)
4      Private uList As List(Of LDUnit)
5      Private lb1OldSelect As Integer
6      Private lb1NewSelect As Integer
7
8      Protected t_unitSerial As String = ""
9      Protected t_unitLNum As String = ""
10     Protected t_unitLocation As String = ""
11     Protected t_unitPhoneNum As String = ""
12     Protected t_unitOwner As LDUnit.Owner
13     Protected t_lockCode As String = ""
14     Protected t_lastDL As DateTime
15
16     Protected t_includeR As Boolean
17     Protected t_includeLC As Boolean
18     Protected t_includeQ As Boolean
19     Protected t_allowCallIns As Boolean
20     Protected t_resetDataYN As Boolean
21     Protected t_resetTimeYN As Boolean
22     Protected t_isEnabled As Boolean
23     Protected t_sendCustP As Boolean
24
25     Protected t_numExceedances As String = ""
26     Protected t_numIntervals As String = ""
27     Protected t_numStartStops As String = ""
28     Protected t_numCalibrations As String = ""
29     Protected t_battVoltage As String = ""
30     Protected t_errorString As String = ""
31     Protected t_excdThreshold As Integer = 0
32     Public t_custParams As String = ""
33     Protected t_calLevel As String = ""
34     Protected t_excdDay As Integer = 0
35     Protected t_excdNight As Integer = 0
36
37
38     Private Sub UnitOptions_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase. ↵
       Load
39         Dim ucomp As New uCompare()
40         Dim comp As Collections.Generic.IComparer(Of LDUnit) = ucomp
41
42         uList = Form1.GetAllUnits()
43
44         'copy ulist to the working list
45         workingUList.AddRange(uList)
46         workingUList.Sort(comp)
47
48         'populate the checked list box
49         Lb1.BeginUpdate()
50         Lb1.Items.Add("All Units")
51         Lb1.Items.Add("Aberdeen")
52         Lb1.Items.Add("CERL")
53
54         For Each unit As LDUnit In workingUList
55             Lb1.Items.Add(unit.UnitNum)
56         Next
57         Lb1.EndUpdate()
58
59         lb1OldSelect = -1
60         lb1NewSelect = -1
61
62         Me.DialogResult = Windows.Forms.DialogResult.None
63         Me.btnChangeExcd.Hide()
64
65         tt.SetToolTip(Me.cbEn, "Indicates whether the program will dial out to the unit." _
66         & vbLf & "Uncheck this box when you don't want the program to call the unit." & vbLf & vbLf & _
67         "To stop a unit from calling in, uncheck ""Allow Unit to Dial In"".")
68         tt.SetToolTip(Me.tbUN, "The ID (""L"") number of the remote unit.")
69         tt.SetToolTip(Me.tbPhone, "The phone number of the remote unit.")
70         tt.SetToolTip(Me.lblC, "The number of self-calibrations performed by the unit since the last data ↵
       reset.")
71         tt.SetToolTip(Me.cbCallIn, "Check this box to make the unit call in." & vbLf & vbLf & _
72         "Uncheck this box to make the unit stop calling in.")
73         tt.SetToolTip(Me.tbExcdDay, "Sets the workday exceedance threshold level (in dB).")
74         tt.SetToolTip(Me.tbExcdNight, "Sets the night and weekend exceedance" & vbLf & "threshold level (in ↵
       dB).")
75         tt.SetToolTip(Me.Button1, "Creates a new unit, using the values entered in this window.")
76         tt.SetToolTip(Me.Button3, "Deletes the currently selected unit.")
77
78     End Sub
79
80
81     Private Sub Lb1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
       Lb1.SelectedIndexChanged
82         lb1NewSelect = Lb1.SelectedIndex
83
84         Select Case lb1OldSelect
85             Case -1 'do nothing
86             Case 0  'write option data to all units in working list
87                 For Each unit As LDUnit In workingUList
88                     'unit.AllowCallIns = t_allowCallIns
```

```vb
 89                            unit.IncludeR = Me.t_includeR
 90                            unit.IncludeQ = Me.t_includeQ
 91                            unit.IncludeLC = Me.t_includeLC
 92                            unit.ResetDataYN = Me.t_resetDataYN
 93                            unit.ResetTimeYN = Me.t_resetTimeYN
 94                            'unit.IsEnabled = Me.t_isEnabled
 95                            unit.SendCustP = Me.t_sendCustP
 96                            If Me.t_sendCustP Then
 97                                unit.CustParams = Me.t_custParams
 98                                Form1.appset.CustParams = Me.t_custParams
 99                            End If
100                        Next
101                        Form1.appset.WorkdayExcdThresh = Me.t_excdDay
102                        Form1.appset.OtherExcdThresh = Me.t_excdNight
103                    Case 1  'write option data to Aberdeen units
104                        For Each unit As LDUnit In workingUList
105                            If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
106                                unit.AllowCallIns = t_allowCallIns
107                                unit.IncludeR = Me.t_includeR
108                                unit.IncludeQ = Me.t_includeQ
109                                unit.IncludeLC = Me.t_includeLC
110                                unit.ResetDataYN = Me.t_resetDataYN
111                                unit.ResetTimeYN = Me.t_resetTimeYN
112                                'unit.IsEnabled = Me.t_isEnabled
113                                unit.SendCustP = Me.t_sendCustP
114                                If Me.t_sendCustP Then
115                                    unit.CustParams = Me.t_custParams
116                                    Form1.appset.CustParams = Me.t_custParams
117                                End If
118                            End If
119                        Next
120                    Case 2  'write option data to CERL units
121                        For Each unit As LDUnit In workingUList
122                            If unit.UnitOwner = LDUnit.Owner.CERL Then
123                                unit.AllowCallIns = t_allowCallIns
124                                unit.IncludeR = Me.t_includeR
125                                unit.IncludeQ = Me.t_includeQ
126                                unit.IncludeLC = Me.t_includeLC
127                                unit.ResetDataYN = Me.t_resetDataYN
128                                unit.ResetTimeYN = Me.t_resetTimeYN
129                                'unit.IsEnabled = Me.t_isEnabled
130                                unit.SendCustP = Me.t_sendCustP
131                                If Me.t_sendCustP Then
132                                    unit.CustParams = Me.t_custParams
133                                    Form1.appset.CustParams = Me.t_custParams
134                                End If
135                            End If
136                        Next
137                    Case Else   'write all form data to the appropriate unit
138                        With workingUList.Item(lb1OldSelect - 3)
139                            .AllowCallIns = t_allowCallIns
140                            .IncludeR = Me.t_includeR
141                            .IncludeQ = Me.t_includeQ
142                            .IncludeLC = Me.t_includeLC
143                            .ResetDataYN = Me.t_resetDataYN
144                            .ResetTimeYN = Me.t_resetTimeYN
145                            .IsEnabled = Me.t_isEnabled
146                            .UnitOwner = Me.t_unitOwner
147                            .UnitPhoneNum = Me.t_unitPhoneNum
148                            .UnitLocation = Me.t_unitLocation
149                            .UnitNum = Me.t_unitLNum
150                            .LockCode = Me.t_lockCode
151                            .SendCustP = Me.t_sendCustP
152                            If Me.t_sendCustP Then
153                                .CustParams = Me.t_custParams
154                            End If
155                            .ExcdDay = Me.t_excdDay
156                            .ExcdNight = Me.t_excdNight
157                        End With
158                End Select
159
160        Select Case lb1NewSelect
161            Case -1 'do nothing
162            Case 0  'gray out Unit Info, write data from the first unit to the form
163                gbUnitInfo.Enabled = False
164                gbStat.Enabled = False
165                cbEn.Enabled = False
166                gbExcd.Enabled = True
167                Me.btnChangeExcd.Show()
168
169                With Me
170                    .t_includeR = workingUList.Item(0).IncludeR
171                    .t_includeQ = workingUList.Item(0).IncludeQ
172                    .t_includeLC = workingUList.Item(0).IncludeLC
173                    .t_allowCallIns = workingUList.Item(0).AllowCallIns
174                    .t_resetDataYN = workingUList.Item(0).ResetDataYN
175                    .t_resetTimeYN = workingUList.Item(0).ResetTimeYN
176                    .t_unitLNum = ""
177                    .t_unitLocation = ""
178                    .t_unitPhoneNum = ""
179                    .t_unitOwner = LDUnit.Owner.Nobody
180                    .t_unitSerial = ""
```

```vb
181                         .t_isEnabled = workingUList.Item(0).IsEnabled
182                         .t_sendCustP = workingUList.Item(0).SendCustP
183                         .t_lockCode = ""
184                         .t_battVoltage = ""
185                         .t_numExceedances = ""
186                         .t_numIntervals = ""
187                         .t_numCalibrations = ""
188                         .t_numStartStops = ""
189                         .t_lastDL = New Date(1900, 1, 1, 0, 0, 0)
190                         .t_errorString = ""
191                         .t_custParams = Form1.appset.CustParams
192                         .t_calLevel = ""
193                         .t_excdDay = Form1.appset.WorkdayExcdThresh
194                         .t_excdNight = Form1.appset.OtherExcdThresh
195                     End With
196                 Case 1  'gray out Unit Info, write data from the first APG unit to the form
197                     gbStat.Enabled = False
198                     gbUnitInfo.Enabled = False
199                     Dim idx As Integer = workingUList.FindIndex(AddressOf IsAberdeen)
200                     cbEn.Enabled = False
201                     gbExcd.Enabled = False
202                     Me.btnChangeExcd.Hide()
203
204                     With Me
205                         .t_includeR = workingUList.Item(idx).IncludeR
206                         .t_includeQ = workingUList.Item(idx).IncludeQ
207                         .t_includeLC = workingUList.Item(idx).IncludeLC
208                         .t_allowCallIns = workingUList.Item(idx).AllowCallIns
209                         .t_resetDataYN = workingUList.Item(idx).ResetDataYN
210                         .t_resetTimeYN = workingUList.Item(idx).ResetTimeYN
211                         .t_unitLNum = ""
212                         .t_unitLocation = ""
213                         .t_unitPhoneNum = ""
214                         .t_unitOwner = LDUnit.Owner.Aberdeen
215                         .t_unitSerial = ""
216                         .t_isEnabled = workingUList.Item(idx).IsEnabled
217                         .t_sendCustP = workingUList.Item(idx).SendCustP
218                         .t_lockCode = ""
219                         .t_battVoltage = ""
220                         .t_numExceedances = ""
221                         .t_numIntervals = ""
222                         .t_numCalibrations = ""
223                         .t_numStartStops = ""
224                         .t_lastDL = New Date(1900, 1, 1, 0, 0, 0)
225                         .t_errorString = ""
226                         .t_custParams = Form1.appset.CustParams
227                         .t_calLevel = ""
228                         .t_excdDay = -1
229                         .t_excdNight = -1
230                     End With
231                 Case 2  'gray out Unit Info, write data from the first CERL unit to the form
232                     gbUnitInfo.Enabled = False
233                     gbStat.Enabled = False
234                     Dim idx As Integer = workingUList.FindIndex(AddressOf IsCerl)
235                     cbEn.Enabled = False
236                     gbExcd.Enabled = False
237                     Me.btnChangeExcd.Hide()
238
239                     With Me
240                         .t_includeR = workingUList.Item(idx).IncludeR
241                         .t_includeQ = workingUList.Item(idx).IncludeQ
242                         .t_includeLC = workingUList.Item(idx).IncludeLC
243                         .t_allowCallIns = workingUList.Item(idx).AllowCallIns
244                         .t_resetDataYN = workingUList.Item(idx).ResetDataYN
245                         .t_resetTimeYN = workingUList.Item(idx).ResetTimeYN
246                         .t_unitLNum = ""
247                         .t_unitLocation = ""
248                         .t_unitPhoneNum = ""
249                         .t_unitOwner = LDUnit.Owner.CERL
250                         .t_unitSerial = ""
251                         .t_isEnabled = workingUList.Item(idx).IsEnabled
252                         .t_sendCustP = workingUList.Item(idx).SendCustP
253                         .t_lockCode = ""
254                         .t_battVoltage = ""
255                         .t_numExceedances = ""
256                         .t_numIntervals = ""
257                         .t_numCalibrations = ""
258                         .t_numStartStops = ""
259                         .t_lastDL = New Date(1900, 1, 1, 0, 0, 0)
260                         .t_errorString = ""
261                         .t_custParams = Form1.appset.CustParams
262                         .t_calLevel = ""
263                         .t_excdDay = -1
264                         .t_excdNight = -1
265                     End With
266                 Case Else   'load data from the selected unit to the class vars and the UI
267                     gbUnitInfo.Enabled = True
268                     gbStat.Enabled = True
269                     cbEn.Enabled = True
270                     gbExcd.Enabled = True
271                     Me.btnChangeExcd.Hide()
272
```

```vb
273                    With Me
274                        .t_includeR = workingUList.Item(lb1NewSelect - 3).IncludeR
275                        .t_includeQ = workingUList.Item(lb1NewSelect - 3).IncludeQ
276                        .t_includeLC = workingUList.Item(lb1NewSelect - 3).IncludeLC
277                        .t_allowCallIns = workingUList.Item(lb1NewSelect - 3).AllowCallIns
278                        .t_resetDataYN = workingUList.Item(lb1NewSelect - 3).ResetDataYN
279                        .t_resetTimeYN = workingUList.Item(lb1NewSelect - 3).ResetTimeYN
280                        .t_unitLNum = workingUList.Item(lb1NewSelect - 3).UnitNum
281                        .t_unitLocation = workingUList.Item(lb1NewSelect - 3).UnitLocation
282                        .t_unitPhoneNum = workingUList.Item(lb1NewSelect - 3).UnitPhoneNum
283                        .t_unitOwner = workingUList.Item(lb1NewSelect - 3).UnitOwner
284                        .t_unitSerial = workingUList.Item(lb1NewSelect - 3).UnitSerial
285                        .t_isEnabled = workingUList.Item(lb1NewSelect - 3).IsEnabled
286                        .t_sendCustP = workingUList.Item(lb1NewSelect - 3).SendCustP
287                        .t_lockCode = workingUList.Item(lb1NewSelect - 3).LockCode
288                        If workingUList.Item(lb1NewSelect - 3).BattVoltage > -1 Then
289                            .t_battVoltage = workingUList.Item(lb1NewSelect - 3).BattVoltage.ToString & " V"
290                        Else
291                            .t_battVoltage = ""
292                        End If
293                        If workingUList.Item(lb1NewSelect - 3).NumExceedances > -1 Then
294                            .t_numExceedances = workingUList.Item(lb1NewSelect - 3).NumExceedances.ToString
295                        Else
296                            .t_numExceedances = ""
297                        End If
298                        If workingUList.Item(lb1NewSelect - 3).NumIntervals > -1 Then
299                            .t_numIntervals = workingUList.Item(lb1NewSelect - 3).NumIntervals.ToString
300                        Else
301                            .t_numIntervals = ""
302                        End If
303                        If workingUList.Item(lb1NewSelect - 3).NumCalibrations > -1 Then
304                            .t_numCalibrations = workingUList.Item(lb1NewSelect - 3).NumCalibrations.ToString
305                        Else
306                            .t_numCalibrations = ""
307                        End If
308                        If workingUList.Item(lb1NewSelect - 3).NumStartStops > -1 Then
309                            .t_numStartStops = workingUList.Item(lb1NewSelect - 3).NumStartStops.ToString
310                        Else
311                            .t_numStartStops = ""
312                        End If
313                        If workingUList.Item(lb1NewSelect - 3).CalLevel > -1 Then
314                            .t_calLevel = workingUList.Item(lb1NewSelect - 3).CalLevel.ToString & " dB"
315                        Else
316                            .t_calLevel = ""
317                        End If
318                        .t_excdThreshold = workingUList.Item(lb1NewSelect - 3).ExcdThreshold
319                        .t_lastDL = workingUList.Item(lb1NewSelect - 3).LastDL
320                        .t_errorString = workingUList.Item(lb1NewSelect - 3).ErrorString
321                        .t_custParams = workingUList.Item(lb1NewSelect - 3).CustParams
322                        .t_excdDay = workingUList.Item(lb1NewSelect - 3).ExcdDay
323                        .t_excdNight = workingUList.Item(lb1NewSelect - 3).ExcdNight
324                    End With
325            End Select
326
327            UpdateForm()
328            lb1OldSelect = lb1NewSelect
329
330        End Sub
331
332        Private Sub UpdateForm()
333
334            cbR.Checked = Me.t_includeR
335            cbQ.Checked = Me.t_includeQ
336            cbLC.Checked = Me.t_includeLC
337            cbCallIn.Checked = Me.t_allowCallIns
338            cbResetD.Checked = Me.t_resetDataYN
339            cbResetT.Checked = Me.t_resetTimeYN
340            cbEn.Checked = Me.t_isEnabled
341            cbCP.Checked = Me.t_sendCustP
342            tbUN.Text = Me.t_unitLNum
343            tbULoc.Text = Me.t_unitLocation
344            tbPhone.Text = Me.t_unitPhoneNum
345            lblSerial.Text = "Serial: " & Me.t_unitSerial
346            lblLock.Text = "Lock Code: " & Me.t_lockCode
347            Select Case Me.t_unitOwner
348                Case LDUnit.Owner.Aberdeen : Me.cbbOwner.SelectedIndex = 0
349                Case LDUnit.Owner.CERL : Me.cbbOwner.SelectedIndex = 1
350                Case LDUnit.Owner.Nobody : Me.cbbOwner.SelectedIndex = 2
351            End Select
352            lblBatt.Text = "Battery Voltage: " & Me.t_battVoltage
353            lblE.Text = "Events: " & Me.t_numExceedances
354            lblI.Text = "Intervals: " & Me.t_numIntervals
355            lblL.Text = "Cals: " & Me.t_numStartStops
356            lblC.Text = "Start/Stops: " & Me.t_numCalibrations
357            If Date.Compare(Me.t_lastDL, New Date(1901, 1, 1, 0, 0, 0)) > 0 Then
358                lblLastDL.Text = "Download Date: " & Me.t_lastDL.ToString("M/d/yyyy h:mm:ss tt")
359            Else
360                lblLastDL.Text = "Download Date: "
361            End If
362            If Lb1.SelectedIndex > 2 Then
363                If Me.t_excdThreshold > -1 Then
364                    lblExcd.Text = "Noise Exceedance Threshold: " & Me.t_excdThreshold.ToString & " dB"
```

```vb
365                 Else
366                     lblExcd.Text = "Noise Exceedance Threshold: "
367                 End If
368             Else
369                 lblExcd.Text = "Noise Exceedance Threshold: "
370             End If
371             lblErr.Text = "Error codes: " & Me.t_errorString
372             lblCal.Text = "CaL Level: " & Me.t_calLevel
373             If Me.t_excdDay < 0 Then
374                 tbExcdDay.Text = ""
375             Else
376                 tbExcdDay.Text = Me.t_excdDay.ToString
377             End If
378             If Me.t_excdNight < 0 Then
379                 tbExcdNight.Text = ""
380             Else
381                 tbExcdNight.Text = Me.t_excdNight.ToString
382             End If
383             If lb1NewSelect > 2 AndAlso (workingUList.Item(lb1NewSelect - 3).CalLevel < 80 OrElse _
384                     workingUList.Item(lb1NewSelect - 3).CalLevel > 110) AndAlso Not _
385                     workingUList.Item(lb1NewSelect - 3).CalLevel = -1 Then
386                 Me.lblCal.BackColor = Color.Red
387                 Me.lblCal.ForeColor = Color.White
388                 Me.tt.SetToolTip(Me.lblCal, "Warning: Calibration level may be outside normal range. " & vbLf & _
        vbLf & "If the unit also has an unrealistic number of exceedances," & vbLf & "the unit may need servicing
        .")
389             Else
390                 Me.lblCal.BackColor = System.Windows.Forms.Control.DefaultBackColor
391                 Me.lblCal.ForeColor = System.Windows.Forms.Control.DefaultForeColor
392                 Me.tt.SetToolTip(Me.lblCal, "The microphone calibration level for the unit.")
393             End If
394         End Sub
395         Private Function IsAberdeen(ByVal unit As LDUnit) As Boolean
396             If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
397                 Return True
398             Else
399                 Return False
400             End If
401         End Function
402         Private Function IsCerl(ByVal unit As LDUnit) As Boolean
403             If unit.UnitOwner = LDUnit.Owner.CERL Then
404                 Return True
405             Else
406                 Return False
407             End If
408         End Function
409
410         Private Sub OK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK.Click
411             'save the information for the current selection
412             Select Case lb1NewSelect
413                 Case -1 'do nothing
414                 Case 0  'write option data to all units in working list
415                     For Each unit As LDUnit In workingUList
416                         'unit.AllowCallIns = t_allowCallIns
417                         unit.IncludeR = Me.t_includeR
418                         unit.IncludeQ = Me.t_includeQ
419                         unit.IncludeLC = Me.t_includeLC
420                         unit.ResetDataYN = Me.t_resetDataYN
421                         unit.ResetTimeYN = Me.t_resetTimeYN
422                         'unit.IsEnabled = Me.t_isEnabled
423                         unit.SendCustP = Me.t_sendCustP
424                         If Me.t_sendCustP Then
425                             unit.CustParams = Me.t_custParams
426                             Form1.appset.CustParams = Me.t_custParams
427                         End If
428                     Next
429                     Form1.appset.WorkdayExcdThresh = Me.t_excdDay
430                     Form1.appset.OtherExcdThresh = Me.t_excdNight
431                 Case 1  'write option data to Aberdeen units
432                     For Each unit As LDUnit In workingUList
433                         If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
434                             unit.AllowCallIns = t_allowCallIns
435                             unit.IncludeR = Me.t_includeR
436                             unit.IncludeQ = Me.t_includeQ
437                             unit.IncludeLC = Me.t_includeLC
438                             unit.ResetDataYN = Me.t_resetDataYN
439                             unit.ResetTimeYN = Me.t_resetTimeYN
440                             'unit.IsEnabled = Me.t_isEnabled
441                             unit.SendCustP = Me.t_sendCustP
442                             If Me.t_sendCustP Then
443                                 unit.CustParams = Me.t_custParams
444                                 Form1.appset.CustParams = Me.t_custParams
445                             End If
446                         End If
447                     Next
448                 Case 2  'write option data to CERL units
449                     For Each unit As LDUnit In workingUList
450                         If unit.UnitOwner = LDUnit.Owner.CERL Then
451                             unit.AllowCallIns = t_allowCallIns
452                             unit.IncludeR = Me.t_includeR
453                             unit.IncludeQ = Me.t_includeQ
454                             unit.IncludeLC = Me.t_includeLC
```

```vb
455                              unit.ResetDataYN = Me.t_resetDataYN
456                              unit.ResetTimeYN = Me.t_resetTimeYN
457                              'unit.IsEnabled = Me.t_isEnabled
458                              unit.SendCustP = Me.t_sendCustP
459                              If Me.t_sendCustP Then
460                                  unit.CustParams = Me.t_custParams
461                                  Form1.appset.CustParams = Me.t_custParams
462                              End If
463                          End If
464                      Next
465                  Case Else   'write all form data to the appropriate unit
466                      With workingUList.Item(lb1OldSelect - 3)
467                          .AllowCallIns = t_allowCallIns
468                          .IncludeR = Me.t_includeR
469                          .IncludeQ = Me.t_includeQ
470                          .IncludeLC = Me.t_includeLC
471                          .ResetDataYN = Me.t_resetDataYN
472                          .ResetTimeYN = Me.t_resetTimeYN
473                          .IsEnabled = Me.t_isEnabled
474                          .UnitOwner = Me.t_unitOwner
475                          .UnitPhoneNum = Me.t_unitPhoneNum
476                          .UnitLocation = Me.t_unitLocation
477                          .UnitNum = Me.t_unitLNum
478                          .LockCode = Me.t_lockCode
479                          .SendCustP = Me.t_sendCustP
480                          If Me.t_sendCustP Then
481                              .CustParams = Me.t_custParams
482                          End If
483                          .ExcdDay = Me.t_excdDay
484                          .ExcdNight = Me.t_excdNight
485                      End With
486          End Select
487
488          'workingUList is returned in the FormClosing event handler
489          Me.DialogResult = Windows.Forms.DialogResult.OK
490          Me.Close()
491      End Sub
492
493      Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel.Click
494          'ulist is returned in the FormClosing event handler
495          Me.DialogResult = Windows.Forms.DialogResult.Cancel
496          Me.Close()
497      End Sub
498
499      Private Sub UnitOptions_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.↵
    FormClosingEventArgs) Handles Me.FormClosing
500          If Me.DialogResult = Windows.Forms.DialogResult.OK Then
501              Form1.ReturnAllUnits(workingUList)
502          Else
503              Form1.ReturnAllUnits(uList)
504          End If
505      End Sub
506
507
508      Private Sub cbR_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cbR.↵
    CheckedChanged
509          If cbR.Checked Then
510              t_includeR = True
511          Else
512              t_includeR = False
513          End If
514      End Sub
515
516      Private Sub cbQ_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cbQ.↵
    CheckedChanged
517          If cbQ.Checked Then
518              t_includeQ = True
519          Else
520              t_includeQ = False
521          End If
522      End Sub
523
524      Private Sub cbLC_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cbLC.↵
    CheckedChanged
525          If cbLC.Checked Then
526              t_includeLC = True
527          Else
528              t_includeLC = False
529          End If
530      End Sub
531
532      Private Sub cbCallIn_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
    cbCallIn.CheckedChanged
533          If cbCallIn.Checked Then
534              Me.t_allowCallIns = True
535          Else
536              Me.t_allowCallIns = False
537          End If
538      End Sub
539
540      Private Sub cbResetD_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
    cbResetD.CheckedChanged
```

```vb
541             If cbResetD.Checked Then
542                 Me.t_resetDataYN = True
543             Else
544                 Me.t_resetDataYN = False
545             End If
546         End Sub
547
548         Private Sub cbResetT_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
            cbResetT.CheckedChanged
549             If cbResetT.Checked Then
550                 Me.t_resetTimeYN = True
551             Else
552                 Me.t_resetTimeYN = False
553             End If
554         End Sub
555
556         Private Sub cbEn_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cbEn.↵
            CheckedChanged
557             If cbEn.Checked Then
558                 Me.t_isEnabled = True
559             Else
560                 Me.t_isEnabled = False
561             End If
562         End Sub
563
564         Private Sub tbUN_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbUN.  ↵
            TextChanged
565             Me.t_unitLNum = tbUN.Text
566         End Sub
567
568
569         Private Sub tbULoc_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbULoc↵
            .TextChanged
570             Me.t_unitLocation = tbULoc.Text
571         End Sub
572
573         Private Sub tbPhone_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles      ↵
            tbPhone.TextChanged
574             Me.t_unitPhoneNum = tbPhone.Text
575         End Sub
576
577         Private Sub cbbOwner_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)    ↵
            Handles cbbOwner.SelectedIndexChanged
578             Select Case cbbOwner.SelectedIndex
579                 Case 0
580                     Me.t_unitOwner = LDUnit.Owner.Aberdeen
581                     Me.t_lockCode = "22222222"
582                 Case 1
583                     Me.t_unitOwner = LDUnit.Owner.CERL
584                     Me.t_lockCode = "22222222"
585             End Select
586         End Sub
587
588         Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.   ↵
            Click
589             'Add a unit to the working list
590
591             Dim newUnit As New LDUnit(Me.t_unitLNum, Me.t_unitPhoneNum, Me.t_unitOwner)
592
593             Me.cbEn.Checked = True 'enables the new unit - usually desired
594
595             With newUnit
596                 .IncludeR = Me.t_includeR
597                 .IncludeQ = Me.t_includeQ
598                 .IncludeLC = Me.t_includeLC
599                 .AllowCallIns = Me.t_allowCallIns
600                 .ResetDataYN = Me.t_resetDataYN
601                 .ResetTimeYN = Me.t_resetTimeYN
602                 .IsEnabled = True 'usually desired when adding a unit
603                 .UnitLocation = Me.t_unitLocation
604             End With
605
606             workingUList.Add(newUnit)
607             Lb1.Items.Add(newUnit.UnitNum)
608
609         End Sub
610
611         Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.   ↵
            Click
612             'Remove a unit from the working list.
613             'MessageBox.Show(Me.lb1NewSelect.ToString)
614             Dim dres As DialogResult
615             Dim sel As Integer = lb1NewSelect
616
617             If lb1NewSelect > 2 Then
618                 dres = MessageBox.Show("Remove unit " & Me.t_unitLNum & "?", "Delete unit", _
619                 MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
620                 'MessageBox.Show(Me.lb1NewSelect.ToString)
621                 If dres = Windows.Forms.DialogResult.Yes Then
622                     Lb1.Items.RemoveAt(sel)
623                     workingUList.RemoveAt(sel - 3)
624                 End If
```

```vb
625              End If
626          End Sub
627
628          Private Sub cbCP_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cbCP.↵
             CheckedChanged
629              If cbCP.Checked Then
630                  Me.t_sendCustP = True
631              Else
632                  Me.t_sendCustP = False
633              End If
634          End Sub
635
636          Private Sub btnCust_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCust.   ↵
             Click
637              Dim cp As New CustomParamsDialog
638              cp.TextBox1.Text = Me.t_custParams
639              Dim dres As DialogResult = cp.ShowDialog()
640              If dres = Windows.Forms.DialogResult.OK Then
641                  Me.t_custParams = cp.TextBox1.Text
642                  cbCP.Checked = True
643              End If
644          End Sub
645
646          Private Sub tbExcdDay_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles    ↵
             tbExcdDay.TextChanged
647              'makes sure the text entered is a number between 0 and 1000
648              Dim res As Integer
649              Dim tryp As Boolean = Integer.TryParse(tbExcdDay.Text, res)
650
651              If tryp = True AndAlso res > 29 AndAlso res < 1000 Then
652                  Me.t_excdDay = res
653                  If tbExcdDay.Text.Length <= 1 Then
654                      Me.cbResetD.Checked = True
655                  End If
656              End If
657          End Sub
658
659
660          Private Sub tbExcdNight_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles   ↵
             tbExcdNight.TextChanged
661              'makes sure the text entered is a number between 0 and 1000
662              Dim res As Integer
663              Dim tryp As Boolean = Integer.TryParse(tbExcdNight.Text, res)
664
665              If tryp = True AndAlso res > 29 AndAlso res < 1000 Then
666                  Me.t_excdNight = res
667                  If tbExcdNight.Text.Length <= 1 Then
668                      Me.cbResetD.Checked = True
669                  End If
670              End If
671          End Sub
672
673          Private Sub btnChangeExcd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles       ↵
             btnChangeExcd.Click
674              'writes day and night exceedance values to all of the units
675
676              For Each unit As LDUnit In workingUList
677                  unit.ExcdDay = Me.t_excdDay
678                  unit.ExcdNight = Me.t_excdNight
679              Next
680
681          End Sub
682      End Class
```

Appendix F: Unit Viewer

```vb
1  Imports System.IO
2
3  Public Class UnitView
4      Private workingUList As New List(Of LDUnit)
5      Private uList As New List(Of LDUnit)
6      Private lb1OldSelect As Integer
7      Private lb1NewSelect As Integer
8      Private lb1NewSelect2 As Integer
9      Private ucomp As New uCompare()
10     Private comp As Collections.Generic.IComparer(Of LDUnit) = ucomp
11     Private currLNum As String
12
13     Protected t_unitSerial As String = ""
14     Protected t_unitLNum As String = ""
15     Protected t_unitLocation As String = ""
16     Protected t_unitPhoneNum As String = ""
17     Protected t_unitOwner As LDUnit.Owner
18     Protected t_lockCode As String = ""
19     Protected t_lastDL As DateTime
20
21     Protected t_includeR As Boolean
22     Protected t_includeLC As Boolean
23     Protected t_includeQ As Boolean
24     Protected t_allowCallIns As Boolean
25     Protected t_resetDataYN As Boolean
26     Protected t_resetTimeYN As Boolean
27     Protected t_isEnabled As Boolean
28     Protected t_sendCustP As Boolean
29
30     Protected t_numExceedances As String = ""
31     Protected t_numIntervals As String = ""
32     Protected t_numStartStops As String = ""
33     Protected t_numCalibrations As String = ""
34     Protected t_battVoltage As String = ""
35     Protected t_errorString As String = ""
36     Protected t_excdThreshold As Integer = 0
37     Public t_custParams As String = ""
38     Protected t_calLevel As String = ""
39
40     Private Sub UnitView_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
41         'load a reference list from disk
42         Try
43             Using fs As New FileStream(Form1.appset.InstallPath & "units.dat", FileMode.Open)
44
45                 Dim bf As New Runtime.Serialization.Formatters.Binary.BinaryFormatter
46                 uList = DirectCast(bf.Deserialize(fs), List(Of LDUnit))
47                 'Deserialize returns an object, which is then cast to a List(Of LDUnit)
48
49             End Using
50         Catch ex As FileNotFoundException   'file does not exist - shouldn't load this form
51             MessageBox.Show("Can't open Unit Viewer - File " & Form1.appset.InstallPath & "units.dat not ↙
    found", _
52             "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
53             Me.btnClose.PerformClick()
54         End Try
55
56         Populate()
57
58         lb1OldSelect = -1
59         lb1NewSelect = -1
60         lb1NewSelect2 = -1
61
62         Me.DialogResult = Windows.Forms.DialogResult.None
63         Me.tsslInProgress.Text = ""
64         Me.tsslUnitsLeft.Text = ""
65
66         tt.SetToolTip(Me.lblEn, "Indicates whether the program will dial out to the unit.")
67         tt.SetToolTip(Me.lblC, "The number of self-calibrations performed by the unit since the last data ↙
    reset.")
68         tt.SetToolTip(Me.lblErr, "The most recent error is at the top of this list.")
69
70         'Me.lblC.BorderStyle = BorderStyle.FixedSingle
71         'Me.lblC.BackColor = Color.Red
72         'Me.lblC.ForeColor = Color.White
73
74
75
76     End Sub
77
78     Private Sub UnitView_Shown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shown
79         Me.RefreshToolStripMenuItem.PerformClick()
80
81     End Sub
82
83     Private Sub Populate()
84         'loads the current LDUnitList in RAM and populates the list box
85         Dim idx As Integer = -1
86         Dim inProgList As New System.Text.StringBuilder()
87
88         'copy contents of current LDUnitList
89         workingUList = Form1.ReadLDUnitList()
90         workingUList.Sort(comp)
```

```vb
 91
 92             'handle the case when units are missing from the current LDUnitList
 93             'because a download is in progress
 94             '(uList is assumed to be the master list, containing one of every unit,
 95             'while workingUList has all of the current data)
 96             For Each u As LDUnit In uList
 97                 Me.currLNum = u.UnitNum
 98                 If Not Me.workingUList.Exists(AddressOf SpecUnit) Then 'unit is checked out
 99                     Dim u2 As LDUnit = u
100                     u2.LastDL = New Date(6, 6, 6)
101                     workingUList.Add(u2)
102                     If inProgList.Length < 1 Then
103                         inProgList.Append(u2.UnitNum)
104                     Else
105                         inProgList.Append(", " & u2.UnitNum)
106                     End If
107                     'else do nothing - the unit is there and more current than the unit in the list on disk
108                 End If
109             Next
110
111             'populate the checked list box
112             workingUList.Sort(comp)
113             Lb1.Items.Clear()
114             Lb1.BeginUpdate()
115             For Each unit As LDUnit In workingUList
116                 Lb1.Items.Add(unit.UnitNum)
117             Next
118             Lb1.EndUpdate()
119
120             If inProgList.Length > 3 Then
121                 Me.tsslInProgress.Text = "Now downloading units " & inProgList.ToString
122             ElseIf inProgList.Length > 0 Then
123                 Me.tsslInProgress.Text = "Now downloading unit " & inProgList.ToString
124             Else
125                 Me.tsslInProgress.Text = ""
126             End If
127
128         End Sub
129
130         Private Function SpecUnit(ByVal unit As LDUnit) As Boolean
131             If String.Compare(unit.UnitNum, Me.currLNum) = 0 Then
132                 Return True
133             Else
134                 Return False
135             End If
136         End Function
137
138         Private Sub Lb1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↵
            Lb1.SelectedIndexChanged
139             lb1NewSelect = Lb1.SelectedIndex
140
141             Select Case lb1NewSelect
142                 Case -1 'do nothing
143                 Case Else   'load data from the selected unit to the class vars and the UI
144                     gbUnitInfo.Enabled = True
145                     gbStat.Enabled = True
146                     With Me
147                         .t_includeR = workingUList.Item(lb1NewSelect).IncludeR
148                         .t_includeQ = workingUList.Item(lb1NewSelect).IncludeQ
149                         .t_includeLC = workingUList.Item(lb1NewSelect).IncludeLC
150                         .t_allowCallIns = workingUList.Item(lb1NewSelect).AllowCallIns
151                         .t_resetDataYN = workingUList.Item(lb1NewSelect).ResetDataYN
152                         .t_resetTimeYN = workingUList.Item(lb1NewSelect).ResetTimeYN
153                         .t_unitLNum = workingUList.Item(lb1NewSelect).UnitNum
154                         .t_unitLocation = workingUList.Item(lb1NewSelect).UnitLocation
155                         .t_unitPhoneNum = workingUList.Item(lb1NewSelect).UnitPhoneNum
156                         .t_unitOwner = workingUList.Item(lb1NewSelect).UnitOwner
157                         .t_unitSerial = workingUList.Item(lb1NewSelect).UnitSerial
158                         .t_isEnabled = workingUList.Item(lb1NewSelect).IsEnabled
159                         .t_sendCustP = workingUList.Item(lb1NewSelect).SendCustP
160                         .t_lockCode = workingUList.Item(lb1NewSelect).LockCode
161                         If workingUList.Item(lb1NewSelect).BattVoltage > -1 Then
162                             .t_battVoltage = workingUList.Item(lb1NewSelect).BattVoltage.ToString & " V"
163                         Else
164                             .t_battVoltage = ""
165                         End If
166                         If workingUList.Item(lb1NewSelect).NumExceedances > -1 Then
167                             .t_numExceedances = workingUList.Item(lb1NewSelect).NumExceedances.ToString
168                         Else
169                             .t_numExceedances = ""
170                         End If
171                         If workingUList.Item(lb1NewSelect).NumIntervals > -1 Then
172                             .t_numIntervals = workingUList.Item(lb1NewSelect).NumIntervals.ToString
173                         Else
174                             .t_numIntervals = ""
175                         End If
176                         If workingUList.Item(lb1NewSelect).NumCalibrations > -1 Then
177                             .t_numCalibrations = workingUList.Item(lb1NewSelect).NumCalibrations.ToString
178                         Else
179                             .t_numCalibrations = ""
180                         End If
181                         If workingUList.Item(lb1NewSelect).NumStartStops > -1 Then
```

```vb
182                             .t_numStartStops = workingUList.Item(lb1NewSelect).NumStartStops.ToString
183                         Else
184                             .t_numStartStops = ""
185                         End If
186                         If workingUList.Item(lb1NewSelect).CalLevel > -1 Then
187                             .t_calLevel = workingUList.Item(lb1NewSelect).CalLevel.ToString & " dB"
188                         Else
189                             .t_calLevel = ""
190                         End If
191                         .t_excdThreshold = workingUList.Item(lb1NewSelect).ExcdThreshold
192                         .t_lastDL = workingUList.Item(lb1NewSelect).LastDL
193                         .t_errorString = workingUList.Item(lb1NewSelect).ErrorString
194                         .t_custParams = workingUList.Item(lb1NewSelect).CustParams
195                     End With
196         End Select
197
198         UpdateForm()
199         lb1OldSelect = lb1NewSelect
200
201     End Sub
202
203     Private Sub UpdateForm()
204
205         If Me.t_includeR Then
206             lblR.Text = "Include Read Parameters: Yes"
207         Else
208             lblR.Text = "Include Read Parameters: No"
209         End If
210         If Me.t_includeQ Then
211             lblQ.Text = "Include Q Parameters: Yes"
212         Else
213             lblQ.Text = "Include Q Parameters: No"
214         End If
215         If Me.t_includeLC Then
216             lblLC.Text = "Include Log and Cal Parameters: Yes"
217         Else
218             lblLC.Text = "Include Log and Cal Parameters: No"
219         End If
220         If Me.t_allowCallIns Then
221             lblCallIn.Text = "Allow Unit to Dial In: Yes"
222         Else
223             lblCallIn.Text = "Allow Unit to Dial In: No"
224         End If
225         If Me.t_resetDataYN Then
226             lblResetD.Text = "Reset Data: Yes"
227         Else
228             lblResetD.Text = "Reset Data: No"
229         End If
230         If Me.t_resetTimeYN Then
231             lblResetT.Text = "Reset Time: Yes"
232         Else
233             lblResetT.Text = "Reset Time: No"
234         End If
235         If Me.t_isEnabled Then
236             lblEn.Text = "Enabled: Yes"
237         Else
238             lblEn.Text = "Enabled: No"
239         End If
240         lblUN.Text = "ID Number: " & Me.t_unitLNum
241         lblULoc.Text = "Location: " & Me.t_unitLocation
242         lblPhone.Text = "Phone: " & Me.t_unitPhoneNum
243         lblSerial.Text = "Serial: " & Me.t_unitSerial
244         lblLock.Text = "Lock Code: " & Me.t_lockCode
245         Select Case Me.t_unitOwner
246             Case LDUnit.Owner.Aberdeen : Me.lblOwner.Text = "Owner: Aberdeen"
247             Case LDUnit.Owner.CERL : Me.lblOwner.Text = "Owner: CERL"
248             Case LDUnit.Owner.Nobody : Me.lblOwner.Text = "Owner: ???"
249         End Select
250         lblBatt.Text = "Battery Voltage: " & Me.t_battVoltage
251         lblE.Text = "Events: " & Me.t_numExceedances
252         lblI.Text = "Intervals: " & Me.t_numIntervals
253         lblL.Text = "Start/Stops: " & Me.t_numStartStops
254         lblC.Text = "Cals: " & Me.t_numCalibrations
255         If Date.Compare(Me.t_lastDL, New Date(1901, 1, 1, 0, 0, 0)) > 0 Then
256             lblLastDL.Text = "Download Date: " & Me.t_lastDL.ToString("M/d/yyyy h:mm:ss tt")
257             lblErr2.Text = "Error Codes" & vbLf & "as of " & vbLf & Me.t_lastDL.ToString("M/d/yy") & " :"
258         ElseIf Me.t_lastDL = New Date(6, 6, 6) Then
259             lblLastDL.Text = "Download Date: In Progress"
260             lblErr2.Text = "Error Codes:"
261         Else
262             lblLastDL.Text = "Download Date: "
263             lblErr2.Text = "Error Codes:"
264         End If
265         If Me.t_excdThreshold > -1 Then
266             lblExcd.Text = "Noise Exceedance Threshold: " & Me.t_excdThreshold.ToString & " dB"
267         Else
268             lblExcd.Text = "Noise Exceedance Threshold: "
269         End If
270         lblCal.Text = "Cal Level: " & Me.t_calLevel
271         If (workingUList.Item(lb1NewSelect).CalLevel < 80 OrElse _
272         workingUList.Item(lb1NewSelect).CalLevel > 110) AndAlso Not _
273         workingUList.Item(lb1NewSelect).CalLevel = -1 Then
```

```vb
274                    Me.lblCal.BackColor = Color.Red
275                    Me.lblCal.ForeColor = Color.White
276                    Me.tt.SetToolTip(Me.lblCal, "Warning: Calibration level may be outside normal range. " & vbLf & ↵
          vbLf & "If the unit also has an unrealistic number of exceedances," & vbLf & "the unit may need servicing↵
          .")
277                Else
278                    Me.lblCal.BackColor = System.Windows.Forms.Control.DefaultBackColor
279                    Me.lblCal.ForeColor = System.Windows.Forms.Control.DefaultForeColor
280                    Me.tt.SetToolTip(Me.lblCal, "The microphone calibration level for the unit.")
281                End If
282
283                'create verbose error codes
284                Dim trimChars3() As Char = {","c}
285                Dim esTemp() As String = t_errorString.Split(trimChars3, StringSplitOptions.RemoveEmptyEntries)
286                Dim errorInts(esTemp.Length - 1) As Integer
287                Dim v As String = ""
288
289                Me.t_errorString = ""
290
291                For i As Integer = 0 To esTemp.Length - 1
292                    Dim tryp As Boolean = Integer.TryParse(esTemp(i), errorInts(i))
293                    If tryp Then 'map a line of text to the numeric code
294                        Select Case errorInts(i)
295                            Case 1 : v = "Out of Memory"
296                            Case 2 : v = "Battery Low"
297                            Case 3 : v = "Power Failure"
298                            Case 4 : v = "Division by Zero"
299                            Case 5 : v = "Operand-1 Range (safe to ignore)"
300                            Case 6 : v = "Operand-2 Range (safe to ignore)"
301                            Case 7 : v = "DPC Format"
302                            Case 8 : v = "Key Has No Effect"
303                            Case 9 : v = "Stop Required (safe to ignore)"
304                            Case 10 : v = "Key Has No Effect In ""View"""
305                            Case 11 : v = "Parameter Entered Wrong (safe to ignore)"
306                            Case 12 : v = "RESET-ALL Required"
307                            Case 13 : v = "Use Arrows, (ON) to Exit"
308                            Case 14 : v = "Use NEXT/PREV or ENTER"
309                            Case 15 : v = "Invalid Numeric Entry"
310                            Case 16 : v = "Open # (hey, I just report it)"
311                            Case 17 : v = "Already Open"
312                            Case 18 : v = "No History Yet"
313                            Case 19 : v = "At End of History"
314                            Case 20 : v = "At Start of History"
315                            Case 21 : v = "History Format Error"
316                            Case 22 : v = "Unknown I/O Command"
317                            Case 23 : v = "I/O Operand Invalid"
318                            Case 24 : v = "Unable to Calibrate"
319                            Case 25 : v = "EEPROM Write Error"
320                            Case 26 : v = "Memory was Lost, Data Reset (check all batteries)"
321                            Case 27 : v = "RECALL- Not Found"
322                            Case 28 : v = "Function Not Implemented"
323                            Case 29 : v = "System Locked"
324                            Case 30 : v = "A:D Stack Full"
325                            Case 31 : v = "A:D Overrun"
326                            Case 32 : v = "RS-232 Framing Error"
327                            Case 33 : v = "RS-232 Line Noisy"
328                            Case 34 : v = "RS-232 Overrun (safe to ignore)"
329                            Case 35 : v = "No Error (truly, a contradiction)"
330                            Case 36 : v = "Power Turned Off"
331                            Case 37 : v = "Time/Date Not Set"
332                            Case 38 : v = "Whatever Old-Ass Printer You Connected to this thing is Busy"
333                            Case 39 : v = "Lithium (backup) Battery Low"
334                            Case 40 : v = "Timer ON Pending"
335                            Case 41 : v = "External Power Failure"
336                            Case 42 : v = "Calibration Changed"
337                            Case 43 : v = "I/O Buffer Overflow"
338                            Case 44 : v = "Crack Overdose - Treatment Needed"
339                            Case 101 : v = "DEVICE ERROR: Count Overflow"
340                            Case 102 : v = "DEVICE ERROR: Exponential Overflow"
341                            Case 103 : v = "DEVICE ERROR: RTX Task Select"
342                            Case 104 : v = "DEVICE ERROR: Illegal Exchange Rate"
343                            Case 105 : v = "DEVICE ERROR: Unknown Interrupt"
344                            Case 106 : v = "DEVICE ERROR: Watchdog Reset"
345                            Case 107 : v = "DEVICE ERROR: RAM Bank Selection Error"
346                            Case 108 : v = "DEVICE ERROR: Opcode Error"
347                        End Select
348                        Me.t_errorString &= errorInts(i).ToString & " - " & v & vbLf
349                    End If
350                Next
351                lblErr.Text = Me.t_errorString
352
353        End Sub
354
355        Private Function IsAberdeen(ByVal unit As LDUnit) As Boolean
356            If unit.UnitOwner = LDUnit.Owner.Aberdeen Then
357                Return True
358            Else
359                Return False
360            End If
361        End Function
362        Private Function IsCerl(ByVal unit As LDUnit) As Boolean
363            If unit.UnitOwner = LDUnit.Owner.CERL Then
```

```vbnet
364                     Return True
365                 Else
366                     Return False
367                 End If
368         End Function
369
370
371 #Region "Menu Strip Items"
372     Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↵
        Handles ExitToolStripMenuItem.Click
373             Me.btnClose.PerformClick()
374         End Sub
375
376     Private Sub SaveToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↵
        Handles SaveToolStripMenuItem.Click
377
378             Me.SaveFileDialog2.InitialDirectory = Form1.appset.InstallPath & "\Logs"
379             Me.SaveFileDialog2.FileName = "Unit Info " & Date.Now.ToString("ddMMMyyyy HHmm")
380             Me.SaveFileDialog2.ShowDialog(Me)
381
382         End Sub
383
384     Private Sub SaveFileDialog2_FileOk(ByVal sender As System.Object, ByVal e As System.ComponentModel.  ↵
        CancelEventArgs) Handles SaveFileDialog2.FileOk
385             'Save the unit data to a CSV file (and eventually to a database table)
386             'first line should be the column headers
387             'subsequent lines should be a row for each unit
388
389             Dim unitLog As New System.Text.StringBuilder()
390             Dim CSB As String = "    <Cell><Data ss:Type=""String"">"
391             Dim CNB As String = "    <Cell><Data ss:Type=""Number"">"
392             Dim CE As String = "</Data></Cell>"
393             Dim aci As String = ""
394             Dim rd As String = ""
395             Dim rt As String = ""
396             Dim r As String = ""
397             Dim q As String = ""
398             Dim lc As String = ""
399             Dim lastDLstr As String = ""
400
401             Using sr1 As New StreamReader(Form1.appset.InstallPath & "sampless_begin.txt")
402                 unitLog.Append(sr1.ReadToEnd())
403             End Using
404
405             unitLog.Append((workingUList.Count + 1).ToString)
406
407             Using sr2 As New StreamReader(Form1.appset.InstallPath & "sampless_middle.txt")
408                 unitLog.Append(sr2.ReadToEnd())
409             End Using
410
411             workingUList.Sort(comp)
412
413             For Each unit As LDUnit In workingUList
414
415                 If unit.LastDL < New Date(1901, 1, 1) Then
416                     lastDLstr = ""
417                 Else
418                     lastDLstr = unit.LastDL.ToString("M/d/yyyy HH:mm:ss")
419                 End If
420                 If unit.AllowCallIns Then
421                     aci = "Yes"
422                 Else
423                     aci = "No"
424                 End If
425                 If unit.ResetDataYN Then
426                     rd = "Yes"
427                 Else
428                     rd = "No"
429                 End If
430                 If unit.ResetTimeYN Then
431                     rt = "Yes"
432                 Else
433                     rt = "No"
434                 End If
435                 If unit.IncludeR Then
436                     r = "Yes"
437                 Else
438                     r = "No"
439                 End If
440                 If unit.IncludeQ Then
441                     q = "Yes"
442                 Else
443                     q = "No"
444                 End If
445                 If unit.IncludeLC Then
446                     lc = "Yes"
447                 Else
448                     lc = "No"
449                 End If
450
451                 unitLog.AppendLine("   <Row>")
452                 unitLog.AppendLine(CSB & unit.UnitNum & CE)
```

```
453                unitLog.AppendLine(CSB & lastDLstr & CE)
454                unitLog.AppendLine(CSB & aci & CE)
455                unitLog.AppendLine(CSB & rd & CE)
456                unitLog.AppendLine(CSB & rt & CE)
457
458                If unit.NumExceedances > -1 Then
459                    unitLog.AppendLine(CNB & unit.NumExceedances.ToString & CE)
460                Else
461                    unitLog.AppendLine(CSB & CE)
462                End If
463                If unit.NumIntervals > -1 Then
464                    unitLog.AppendLine(CNB & unit.NumIntervals.ToString & CE)
465                Else
466                    unitLog.AppendLine(CSB & CE)
467                End If
468                If unit.NumStartStops > -1 Then
469                    unitLog.AppendLine(CNB & unit.NumStartStops.ToString & CE)
470                Else
471                    unitLog.AppendLine(CSB & CE)
472                End If
473                If unit.NumCalibrations > -1 Then
474                    unitLog.AppendLine(CNB & unit.NumCalibrations.ToString & CE)
475                Else
476                    unitLog.AppendLine(CSB & CE)
477                End If
478
479                If unit.ExcdThreshold > -1 Then
480                    unitLog.AppendLine(CNB & unit.ExcdThreshold.ToString & CE)
481                Else
482                    unitLog.AppendLine(CSB & CE)
483                End If
484                If unit.CalLevel > -1 Then
485                    unitLog.AppendLine(CNB & unit.CalLevel.ToString & CE)
486                Else
487                    unitLog.AppendLine(CSB & CE)
488                End If
489                If unit.BattVoltage > -1 Then
490                    unitLog.AppendLine(CNB & unit.BattVoltage.ToString & CE)
491                Else
492                    unitLog.AppendLine(CSB & CE)
493                End If
494
495                unitLog.AppendLine(CSB & unit.ErrorString & CE)
496                unitLog.AppendLine(CSB & r & CE)
497                unitLog.AppendLine(CSB & q & CE)
498                unitLog.AppendLine(CSB & lc & CE)
499                unitLog.AppendLine("   </Row>")
500
501            Next
502
503            Using sr3 As New StreamReader(Form1.appset.InstallPath & "sampless_end.txt")
504                unitLog.Append(sr3.ReadToEnd())
505            End Using
506
507            Using sw1 As New StreamWriter(Me.SaveFileDialog2.FileName)
508                sw1.Write(unitLog.ToString())
509            End Using
510
511        End Sub
512        Private Sub RefreshToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  ↵
           Handles RefreshToolStripMenuItem.Click
513            'stuff to do to refresh the display
514            'display is refreshed whenever a unit is checked in or out of the LDUnitList
515
516            lb1NewSelect2 = Lb1.SelectedIndex
517            Populate()
518            If lb1NewSelect2 > -1 Then
519                Lb1.SelectedIndex = lb1NewSelect2
520            End If
521
522        End Sub
523
524 #End Region
525
526
527        Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClose.  ↵
           Click
528            Me.DialogResult = Windows.Forms.DialogResult.Cancel
529            Me.Close()
530        End Sub
531
532        Private Sub UnitView_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.              ↵
           FormClosingEventArgs) Handles Me.FormClosing
533            Form1.IsUVOpen = False 'note that this writes to a _property_, not a field
534
535            If e.CloseReason = CloseReason.UserClosing OrElse e.CloseReason = CloseReason.None Then
536                'cancel the form closing and disposal - instead, just hide the form
537                e.Cancel = True
538                Me.Hide()
539            End If
540            'don't return (add) any units here to the LDUnitList
541        End Sub
```

```
542
543 End Class
```

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 05-2007	2. REPORT TYPE Final	3. DATES COVERED (From - To)
4. TITLE AND SUBTITLE ERDC-CERL LD-870 Download Program: Programming Manual		**5a. CONTRACT NUMBER** MIPR6FXXR3A563
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Ben Niemoeller and Edward T. Nykaza		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER 0B72D9
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center (ERDC) Construction Engineering Research Laboratory (CERL) PO Box 9005 Champaign, IL 61826-9005		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CERL SR-07-7
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Commander, U.S. Army Aberdeen Test Center 400 Colleran Road Aberdeen Proving Ground, MD 21005-5059		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The U.S. Army Engineer Research and Development Center Construction Engineering Research Laboratory has developed software that interfaces with an array of Larson-Davis Model 870 Environmental Noise Monitors for Aberdeen Test Center. This document explains logic and procedures used while programming the software that are of interest to a programmer looking to modify or expand the functionality of the program. The following topic areas are covered: terminology, time synchronization, and scheduling events. This document will be of interest to those who wish to modify the L-D Download software program. The code, which was written with Microsoft Visual Studio 2005, is included in the appendices.

15. SUBJECT TERMS
Programming manual, noise monitor, military training, Aberdeen Test Center, MD, environmental management, data management

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR		**19b. TELEPHONE NUMBER (include area code)**